;login:

THE MAGAZINE OF USENIX & SAGE

April 2000 • volume 25 • number 2

USENIX celebrates its 25th anniversary at the 2000 Annual Technical Conference in San Diego! Join the party!

See the full program on page 82

inside:

USENIX & SAGE

The Advanced Computing Systems Association & The System Administrators Guild

USENIX Upcoming Events

SANE 2000—2nd International System Administration and Networking Conference

Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet

MAY 22-25, 2000
Maastricht, The Netherlands
http://www.nluug.nl/events/sane2000/

2000 USENIX Annual Technical Conference

JUNE 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
http://www.usenix.org/events/usenix2000

3rd Large Installation System Administration of Windows NT/2000 Conference (LISA-NT 2000)

JULY 30 - AUGUST 2, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
http://www.usenix.org/events/lisa-nt2000

4th USENIX Windows Systems Symposium

AUGUST 3-4, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
http://www.usenix.org/events/usenix-win2000

9th USENIX Security Symposium

Sponsored by USENIX in cooperation with the CERT Coordination Center

AUGUST 14-17, 2000
Denver Marriott City Center, Denver, Colorado, USA
http://www.usenix.org/events/sec2000

4th Annual Linux Showcase and Conference

Sponsored by USENIX and Atlanta Linux Showcase, Inc., in cooperation with Linux International

OCTOBER 10-14, 2000
Atlanta, Georgia, USA
http://www.linuxshowcase.org
Extreme Linux submissions due: April 17, 2000
Hack Linux/Use Linux submissions due: May 1, 2000

The First Workshop on Industrial Experiences with Systems Software (WIESS'2000)

Co-sponsored by IEEE TCOS, and ACM SIGOPS (pending approval)

OCTOBER 22, 2000
Co-located with the 4th Symposium on Operating Systems Design & Implementation (OSDI 2000), Paradise Point Resort, San Diego, CA
http://www.usenix.org/events/osdi2000/wiess2000
Submissions due: May 15, 2000

4th Symposium on Operating System Design & Implementation (OSDI 2000)

Co-sponsored by IEEE TCOS and ACM SIGOPS

OCTOBER 23-25, 2000
Paradise Point Resort, San Diego, California, USA
http://www.usenix.org/events/osdi2000
Submissions due: April 25, 2000

14th Systems Administration Conference (LISA 2000)

Sponsored by USENIX & SAGE

DECEMBER 3-8, 2000
New Orleans, Louisiana, USA
http://www.usenix.org/events/lisa2000
Submissions due: June 6, 2000

6th USENIX Conference on Object-Oriented Technologies and Systems

JANUARY 29 - FEBRUARY 2, 2001
San Antonio, Texas, USA
http://www.usenix.org/events/coots01
Submissions due: July 27, 2000

Java Virtual Machine Research and Technology Symposium

APRIL 23-24, 2001
Monterey, California, USA

For a complete list of future USENIX events, access **http://www.usenix.org/events**

contents

motd

by Rob Kolstad

Dr. Rob Kolstad works as a program manager organizing computer security conferences. Longtime editor of *;login:*, he is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

In Which Rob Takes a Break

In my avocations (and vocation), I often ask people if they can assist with some task or if they wish to exploit some opportunity. It is not unusual for them to say, "Oh no, there's just not enough time for me to do that."

Of course, except during the edge conditions of birth and death, we are all allocated a uniform 168 hours/week. How we spend those hours is a matter of priorities. The message-behind-the-message is "That idea/task is just not high enough on my priority list." That's a fine message! Taking on too much is a stressful thing that is not normally a healthy activity in the long run.

I try to do my share of responsibility-accepting, what with the USA Computing Olympiad and the regional science fair (not to mention USENIX). Lately, I've been asking myself to complete various tasks around the house like: complete the remote control system for the new house, clean up the photo darkroom, finish the fancy little digital mixers for the in-house sound system, construct a video switcher interface, plant a garden, trim some trees over at the edge of the lawn, test some new grass seeds to see if they'll grow at this elevation, fancy up the game-show hardware and software, and so on. All these projects are running one to two years late!

Now, if I think it's important enough to ask myself to do something, you'd think such tasks would be important enough for me to accept! And I really think they really are important. I just don't have the time (oops!).

Obviously, making that same excuse exposes a priority issue. So I checked my priorities.

I spend a lot of time working.

If I could just eliminate working, I'd have all sorts of extra time to do those (other) important tasks. But there's the money problem.

So, I scrimped and saved last year and have squirreled away enough cash that I can take 6–12 months off from work so I can finish the house projects and others to get my life in order. I've even taken up swimming and racketball again to fix my personal health (vs. my schedule health).

By the time you read this, I should be attacking the backlog to see if spending time at work really was the time sink it appears to be. I've laid in all the tools I think I'll need (including electronics tools, a skid loader, seeds, etc.). It should be an interesting experiment.

I've talked to several people about this idea. Their support has been unanimous, ranging from, "Oh yeah, I do that every seven years," to "Great idea! Why don't I do that, too?"

I don't know if it's a good idea or not. I'll let you know how the experiment goes.

apropos

For the Love of the Game

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a consultant on Internet firewalls and network connections and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

Michael Jordan is well known for many things: defying gravity, selling tennies, and trying baseball are just a few. In the world of sports contracts, he's also known for breaking ground by adding a clause to his contract with the Bulls that allowed him to continue to play basketball when and where he wanted to just "for the love of the game." Translated from legalese, he was allowed to do what he did for a living for free if he wanted to. That clause meant he could participate in any "pick-up" game on any blacktop of his choosing just because he loved to do it, no strings attached, and no one could say that he couldn't.

What does a computer professional have in common with a basketball star? Well, both work for a living. Jordan does what he does best and gets paid (really well!) for it. And we computer professionals do what we do for a living and get paid (pretty well!) for it.

Now I know that, as far as money is concerned, it's probably safe to speculate that Jordan could have retired long ago, but he kept on working. "Yeah, right," I can hear you say. "Playing a game doesn't count as work." But I'm sure, when it came down to it, there were some days that he didn't want to hit the gym any more than any of us want to hit the keyboards. We all have those days. But, while we do what we do to make money, more important, we do it because we like it.

Where I live, in the Silicon Valley, becoming a millionaire isn't out of the realm of possibility. There are an awful lot of start-ups to choose from and plenty of examples of those who have made it big. I often quip that you can see more exotic cars in the parking lot of a successful start-up than you can at an organized car show, and I'm only half kidding. This kind of within-reach wealth encourages folks to entertain thoughts of what they'd do if they didn't have to work anymore, and it's a frequent topic of lunchtime conversation at the local restaurants.

Common financial-independence fantasies include dream vacations, retirement locations, and philanthropic work. Perhaps surprisingly, many folks also make comments like, "I'd still work." Sure, they may not work as hard, or as much, but they often say they'd still work in some capacity. That leads me to conclude that we techies work for a living, but we also work because we love it, and that's just like Jordan.

So, it takes more than just money to motivate workers. That's not new news. Or is it? In this economic environment, talented computer professionals are hot commodities and no employer can afford to lose good people, yet I see it happen all the time. It appears that some in management fail to understand this: at some level, no amount of money will make up for an oppressive work environment.

The bottom line is that compensation is equally as important as nonmonetary considerations such as opportunity for professional growth, input to decisions, flexible hours, pro bono work, fundamental fairness, and respect. Successful managers find out what is important to their employees and partner with them to create an environment which is good both for the organization and for the employees. This keeps the game fun for everyone.

Maybe the Bulls didn't care if Michael Jordan played hoops with old high school friends, or with his kids at a park. Maybe he wouldn't have walked away from the job over the clause that allowed him to do so. Smart move on the Bulls' part that they didn't push the issue to find out, because when it comes down to it, most of us work partly because we have to, and partly for the love of the game.

;login:

;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published eight times a year by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$40 of each member's annual dues is for an annual subscription to ;login:. Subscriptions for nonmembers are $50 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to ;login:, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

EDITORIAL STAFF

EDITORS:
Tina Darmohray <tmd@usenix.org>
Rob Kolstad <kolstad@usenix.org>

STANDARDS REPORT EDITOR:
David Blackwood <dave@usenix.org>

MANAGING EDITOR:
Jane-Ellen Long <jel@usenix.org>

COPY EDITOR:
Eileen Cohen

TYPESETTER:
Festina Lente

MEMBERSHIP AND PUBLICATIONS
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org>

The closing dates for submission to the next two issues of ;login: are 5 April 2000 and 2 May 2000.

letters to the editor

APOLOGIES

Our heartfelt apologies to Josh Simon of Collective Technologies, who coordinated the summaries for LISA '99 (Feb. 2000 issue). In addition to coordinating and also writing summaries of this year's LISA, Josh wrote the summaries of the Advanced Topics Workshop and Dan Klein's invited talk at LISA '98 (Feb. '99 issue). We mistakenly attributed those summaries to Bruce Alan Wynn, the coordinator of last year's summarizers.

THE EDITORS

ON UNIX CERTIFICATION

FROM BENJY FEEN
<benjy@monkeybagel.com>

[*Editor's note: This note went by on a mailing list recently. It was some of the most interesting commentary on certification I've read in a long time. – RK*]

My UNIX certification rant: When I worked for a certain company that made Incredibly Big Machines, we were encouraged to get certified, and people often hung the test scores, not the certificates, on the wall, out of pride. It was there that I learned to value certification as a form of continuing education.

Me, I learned what I know about UNIX by spending most of what I earned at my student job on every O'Reilly I could find. I even owned the Curses O'Reilly, for god's sake. And then I'd love to read those books. It was really disturbing to get into the big world and find out that only one of my coworkers read tech references on his own time. But when my teammates went for a cert, they'd hit the books, and the result was that they'd gain in knowledge they'd never have sought out for its own sake. Point being that if someone gets a certification, it's at least arguable that they're trying, and if they seek to obtain the knowledge of a more experienced admin, so much the better.

Of course, it's probably wise to be wary of people who feel the need to make gratuitous references to their certifications (or college degrees, for that matter). I don't blame anyone for getting sick of paper tigers, though; I worked with a fully pedigreed MCSE who was unable to set the system clock on his servers.

Regarding NT certifications: It's unfortunate, but not everyone in the extended family of systems administration is a big geek. NT does not attract big geeks. Big geeks like cool tech and nifty puzzles to solve. NT is not cool tech, and among the many, many things NT is, it is not, never has been, and will likely never be nifty.

My experience is that NT admins tend to be bright folks who decided to bootstrap themselves into a computer career. For people who are looking for a career but don't yet have a Clue, certification is a well-paved eight-lane highway with a brightly lit on-ramp. This highway doesn't lead directly to a Clue, but there's an unmarked access road if you know where to look.

Most of the time, the wayward novice just needs a friendly shove off the road. And of course, lots of people never get a Clue and believe that the highway is all there is. Luckily, these folks are driving around in bright orange Dodge Challengers (license: GOTMCSE) with glasspack exhausts and are easy to spot and avoid.

more letters to the editor

IN DEFENSE OF FREEBSD

FROM STEN DRESCHER
<stend@support.tivoli.com>

In his letter in the February 2000 *;login:*, David Maxwell stated that "I looked today at *<http://www.freebsd.org>* and *<http://www.linux.org>*, and I couldn't find any mention of a non-Intel installation." I don't see why he expected to find a mention at *<http://www.freebsd.org>*, since, as Mr. Maxwell stated earlier in his letter, "The FreeBSD folk wanted to focus on the i386 platform and put their full energy into it." The FreeBSD Web site itself says much the same thing: "FreeBSD is an advanced BSD UNIX operating system for 'PC-compatible' computers." As for Linux, Mr. Maxwell apparently didn't look too hard, since one of the prominent links at *<http://www.linux.org>* is "Can I run Linux on my type of computer?" The answer is at *<http://www.linux.org/help/beginner/platforms.html>*:

> Linux runs successfully on most computers, laptops, and platforms. There are several projects underway to port Linux to other hardware configurations. An overview of hardware compatibility resources are listed below:
>
> Supported PC-based CPUs include:
> Intel/AMD/Cyrix 386SX/DX/SL/DXL/SLC
> Intel/AMD/Cyrix486SX/DX/SL/SX2 /DX2/DX4
> AMD K5, K6, K6-2, K6-3 and K7/Athlon Cyrix 6x86, 6x86MX
> Intel Pentium, Pentium Pro, Pentium II (including the Celeron series) and Pentium III
> IDT WinChip C6
> Symmetrical Multiprocessing (multiple CPUs)
> more . . .
>
> Supported non–PC-based platforms include:
> Digital Alpha
> Sun SPARC
> Macintosh
> more . . .

That link has been there for several months, as I used it when searching for information about possible operating systems for a Sparc ELC. Beyond that, *linux.org* is not the "official" site for Linux, the way *freebsd.org* is for FreeBSD. The closest thing to an official site for Linux would be *<http://www.kernel.org>*, "the primary site for the Linux kernel source." Finding mention of non-Intel Linux is even more trivial there, as it is on the front page.

Later, Mr. Maxwell states that "Linux and FreeBSD may claim that they support multiple platforms, but the code is not integrated into their source at this time." As I've already mentioned, FreeBSD makes no such claim. As for Linux, the statement is false – the base kernel source has included the Alpha, Motorola 68K, MIPS, PowerPC, and Sparc code since the 2.0 days; 2.2 added ARM and Sparc64; and the 2.3 kernels add the IA-64 and SuperH processors. Perhaps the S/390 code qualifies as "not integrated into their source" – S/390 support appears in the 2.2.14 kernel but not yet in the 2.3 series.

I certainly agree with Mr. Maxwell's closing statement that NetBSD should have received more attention, but it isn't necessary to spread falsehoods and innuendo about other operating systems to make a "sales pitch" for his preferred choice. More to the point, I expect the *;login:* editors to do at least some rudimentary fact checking before accepting letters for publication.

Rik Farrow, who wrote the article that inspired this exchange, responds:

We do not fact-check letters to the editor unless the letter contains information that is either obviously or logically untrue. We simply provide a forum within which a writer can make a point (or embarrass him- or herself).

WINDOWS NT

FROM BILL TROST
<trost@cloud.rain.com>

Reading over the article "Building a Windows NT Bastion Host" in the February issue of *;login:*, I was appalled by the opening sentences: "This article . . . makes little or no attempt to explain or discuss . . . " I was supposed to be reading "The Magazine of USENIX and SAGE," not "Cookbook NT Administration"!

Until now, I have not been particularly interested in the discussion about whether USENIX should be paying attention to NT, but upon seeing what such attention is doing to the Association, I am now convinced we would be much better off limiting our focus on NT to its interaction with UNIX systems. There are plenty of organizations related to administering NT itself; there is little point in USENIX becoming yet another one.

speaking HTTP

by Oleg Kiselyov

Oleg Kiselyov is a computer scientist with Computer Sciences Corporation, in Monterey, California. Soon it will be twenty years since he started using computers to solve somebody else's problems.

<oleg@pobox.com>

A File-Uploader Tool

This article describes a simple HTTP uploading tool. HTTP is commonly viewed as something that happens between a browser and a Web server. However, HTTP is useful in its own right, for example, as a good file-distribution protocol with a number of important advantages over ftp. This article gives an example how to speak HTTP and get understood.

The HTTP uploader is somewhat reminiscent of Microsoft Frontpage's server extensions. It lets you push (binary or text) content – Web pages, images, binary files – from one computer to another. If the source platform is Win*xx*/WinNT, you can make a shortcut of a script that will let you upload files just by dragging and dropping them onto an icon. The tool works through Web proxies and gateways. If you can download Web pages, you should be able to upload files as well.

The uploader tool works on various versions of UNIX and WinNT/Win*xx* with different HTTP servers: I tried Apache, Netscape, and IIS. The tool is made of two Perl scripts, one of them being a CGI script. The choice of the implementation language is accidental and irrelevant. What deserves admiration is the HTTP protocol, whose power and simplicity make even far more complex applications possible.

HTTP Protocol

By definition[1], HTTP is a request/response protocol that exchanges messages in a format similar to that used by Internet mail (MIME). An HTTP transaction is essentially a remote procedure call. It is usually a blocking call, although HTTP/1.1 provides for asynchronous and batch modes. HTTP allows intermediaries (caches, proxies) to cut into the response-reply chain.

An operation to execute remotely is expressed in HTTP as an application of a request method to a resource. Additional parameters, if needed, are communicated via request headers or a request body. The request body may be an arbitrary octet-stream. The HTTP/1.1 standard defines methods GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE, and CONNECT. A particular server may accept many others. This extensibility is a rather notable feature of HTTP. The parties can use not only custom methods but custom request and reply headers as well. In addition, a client and a server may exchange meta-information via "name=value" attribute pairs of the standard "Content-Type:" header.

Most of the HTTP transactions performed every day are done behind the scenes by browsers, proxies, robots, and servers. Yet the protocol is so simple that one can easily speak it oneself. The only requirement is a language or tool that is able to manipulate text strings and establish TCP connections. Even a simple telnet application may do in a pinch, which is often useful for debugging. Server-side programming is less demanding: a servlet or a scriptlet does not need to bother with the network connectivity, authentication, access restrictions, SSL, and other similar chores. Server modules or FastCGI give a server-side programmer even more tools: load-balancing, persistence, database connectivity, etc. This article demonstrates how to use Perl scripts to speak and respond HTTP directly.

Making an Upload Request – An uptow Script

An uptow is a Perl script that speaks the client part of HTTP. It asks a server to perform a remote operation: store submitted data in a desired location. The server will respond with the result code or an error message. The script is called as follows:

```
uptow dest-directory local-file-path
```

It will copy the file specified by local-file-path to a remote site. The data will be placed into a specified dest-directory on the remote site under the same (base) filename. The server will typically prepend a predefined path to this dest-directory (e.g., /usr/local/htdocs or /w/data) to confine file updates to that part of its file system. This script publishes the files synchronously and always tells the result of the transfer.

The remote site to which to publish is identified by a number of configurational parameters: $REMOTE_HOST, $REMOTE_PORT, and $TAKER_URI. It is trivial to modify the script to get these parameters from environment variables or to read them from a configuration file.

When called as uptow mysite/dev /tmp/data.txt, the script establishes a TCP connection to a destination HTTP server ($REMOTE_HOST) and sends the following message:

```
PUT /cgi-bin/admin/Update-w-Taker.pl/mysite/dev/ HTTP/1.0 CRLF
Host: hostname.org:80 CRLF
User-Agent: UPTOW/1.3 CRLF
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== CRLF
Content-Type: text/plain; filename="data.txt" CRLF
Content-Length: 1234 CRLF
CRLF
contents of the file /tmp/data.txt as it is.
```

The first line of the message is a request line. It is followed by request headers, in a "Name: value" format similar to that of RFC822 mail headers. The names in the headers are case-insensitive. The first empty line signifies the end of the headers. If the message includes a body – as it does in our case – the payload data is sent immediately after the empty line. The request line and the header lines are terminated by a carriage return/line feed (CRLF) character sequence: the character with decimal code 13, followed by the character with decimal code 10.

The request line tells what operation to perform, where to put the payload data, and what version of the HTTP protocol we will speak. Some obsolete firewalls and proxies may either refuse PUT requests outright or break the connection without indicating any error. Also, some Web servers may be configured by default to disallow PUT methods (see below for the server configuration). If you encounter that situation, you can change the uptow and Update-w-Taker.pl scripts to use a POST method instead. The latter is more widely accepted. The PUT method nevertheless seems to be the most appropriate for upload. The location to store the payload data is specified by a Uniform Resource Identifier (URI). It is a character string that looks like an absolute UNIX file path. The meaning may, however, be different, as we will see. The uptow script creates this URI by appending the desired upload location to the $TAKER_URI. The "update taker" section below explains what an HTTP server does with such a string.

It may happen that the server to which we upload data is not directly accessible. For example, the server and the uptow client may be separated by a firewall. All HTTP transactions between computers on the different sides of a firewall must therefore go through a dedicated Web proxy or a gateway. A Web browser or any other HTTP client has to be made aware of such an arrangement. Specifically, to tell the uptow script to use a proxy you have to set a $PROXY_NAME parameter. The script will then connect to that proxy and have it relay the request to the destination server. The relay request looks just like the direct upload request above. Only the first line is slightly different:

The request line tells what operation to perform, where to put the payload data, and what version of the HTTP protocol we will speak.

The *uptow* script tries to check whether a file to send is ASCII or binary. If it's ASCII, the media type of the Content-Type header is set to "*text/plain,*" and the file is sent as it is. Otherwise, the data is encoded into a hexadecimal stream.

```
PUT http://hostname.org/cgi-bin/admin/ Update-w-Taker.pl/mysite/dev/
            HTTP/1.0 CRLF
```

That is, instead of a URI naming a resource to create, we send the full URL, including the "http://hostname.org" part. Here hostname.org is the name of a host to which we upload the file. The proxy strips away this "http://hostname.org" part when it sends the request to the destination server.

The HTTP protocol defines a number of headers that should or may be used during an HTTP exchange. Here we will describe a particular subset of headers that is used in file-upload transactions.

The Host header identifies the request target server. It is a good idea always to supply this header. Moreover, it is mandatory in version 1.1 of the HTTP protocol. The User-Agent header identifies the client software – the uptow script, in our case. The server usually quotes this information in its logs. An HTTP server may be configured to demand to know the identity of a user itself before it will consider a request of its agent. The Authorization header should be present then to specify an authentication scheme and the corresponding credential. In the most basic authentication scheme – the one used in the example shown – a user is identified by a symbolic ID and the password. These two strings separated by a single colon (:) character and BASE64-encoded constitute the corresponding credential. Every Web server is guaranteed to support the basic scheme. Yet it is hardly secure, since it transmits passwords in an easily decodable form – almost in plain text. Incidentally, ftp and telnet protocols suffer the same problem. HTTP/1.1 defines a more secure Digest scheme[2]. An HTTP client may also attempt a Secure Socket Layer (SSL) connection to a Web server. SSL is a lower-level (transport) protocol; therefore, the content of an HTTP conversation is unaffected by the fact it is to be transmitted over an SSL connection.

When a request such as ours has a body, the type and the size of its data have to be identified, by Content-Type and Content-Length headers. The former should tell the media type of the data: the "MIME type," as it is often called. Content-Length is the size of the body in bytes. If the data being uploaded is ASCII text, the media type may be set to "text/plain," as above. When the payload is intended to be stored without any further processing, the "application/octet-stream" MIME type seems the most appropriate. Although the request line and the headers are in ASCII, the body of an HTTP message can carry arbitrary data.

Unfortunately, some obsolete Web proxies and gateways (notably Raptor 5.0) are not 8-bit transparent: they do not like zero bytes in a request stream. Apparently firewall programmers used a function strncpy() where memcpy() would have been more appropriate. The uptow script tries to check whether a file to send is ASCII or binary. If it's ASCII, the media type of the Content-Type header is set to "text/plain," and the file is sent as it is. Otherwise, the data is encoded into a hexadecimal stream; BASE64 encoding can be used as well. The media type of "application/x-octet-stream-b2a" identifies the encoded content. A Transfer-encoding header may seem the most fitting place to specify an encoding. Alas, Apache accepts only one value for this request header: chunked. Any other value in Transfer-encoding results in a BAD_REQUEST error. In any case, the payload encoding concerns only pushing of data via a particular obsolete proxy, which I happen to be burdened with. If your Web proxy follows the HTTP standard or you connect to a server directly, you can set the media type to "text/plain" or "application/octet-stream" and forget about encoding.

It is not commonly recognized that a Content-Type header may carry parameters in the "name=value" format. The parameters are separated from the media type and from one another by semicolons. The value can be an arbitrary string, possibly quoted if it contains spaces and other special characters. In our example, Content-Type has one parameter: filename. It tells the base name of the file being uploaded. We could have just as well passed this information via a custom request header, for example, X-Filename: data.txt. HTTP is an extensible protocol, which explicitly allows custom headers. A server ignores any headers it does not recognize.

HTTP protocol has another powerful feature that unfortunately remains relatively obscure: the body of an HTTP message may be composed of several parts.

HTTP protocol has another powerful feature that unfortunately remains relatively obscure: the body of an HTTP message may be composed of several parts. This is similar to multipart/mixed or multipart/digest MIME email messages, which may carry several pieces of information within a single entity. We can therefore upload several files in one transaction by encapsulating them as separate parts of a single request body. We can also upload a tar file and have the server extract its members. In any case, the corresponding modifications to the uptow and Update-w-Taker.pl scripts are trivial.

Strictly speaking, we do not have to use the uptow script to upload a file. For example, we can forego convenience and employ a spartan tcp-transaction tool, tcp-trans[3]. We can even enter telnet hostname.org 80 on the command line and type in the request, line by line. Pressing a "Return" key is enough to terminate a line. Although it is not the same as sending the CRLF combination, many Web servers are rather forgiving.

If a server accepts the submitted data and successfully stores it in a desired location, it sends an acknowledgment, an HTTP message:

```
HTTP/1.1 201 Created /w/data/mysite/dev/data.txt CRLF
Server: Apache/1.3.6 (Unix) CRLF
Date: Fri, 29 Oct 1999 00:18:48 GMT CRLF
CRLF
```

The first line of a server response is a status line. It tells the protocol version the server speaks, a numerical result code, and a brief description of the success or failure of the request. The numerical code is a three-digit number intended primarily for a nonhuman agent. A code within the 200 range signifies a successful completion of a request. A 3*xx* result code tells the agent that an additional action is necessary; a 4*xx* code is returned if the request is invalid or cannot be fulfilled (for example, because a user failed to authenticate itself or does not have sufficient permissions). Result codes within the 500 range indicate a serious problem on the server side; see the HTTP document[1] for more details.

The status line in a server response is followed by reply headers and an empty line. The latter signifies the end of the headers. The response body, if sent, follows right after the empty line. In case of the 201 reply, there is no body. If the server rejects an upload request or fails to satisfy it, the server sends a response as well, with an appropriate error code:

```
HTTP/1.1 403 Forbidden CRLF
Server: Microsoft-IIS/4.0 CRLF
Date: Tue, 02 Nov 1999 16:55:00 GMT CRLF
Content-type: text/html CRLF
CRLF
<title>THW-taker Error</title>
<h1>THW-taker Error</h1>
This server encountered an error:<p> <b> d:/temp/bb/uptow.pl is not writable, No
such file or directory </b>
```

All but the well-known request headers are passed as environment variables whose names start with "HTTP_", e.g., HTTP_HOST and HTTP_USER_AGENT.

Update Taker – An Uploading Server

Update-w-Taker.pl is a CGI script to update a Web site remotely. It takes submitted data sent by the uptow script or a similar application and stores the data in a desired place within the $Dest_root directory tree.

When an HTTP daemon receives the request, the daemon notices that the request URI string starts with /cgi-bin/. This matches a ScriptAlias rewriting template of the server's configuration. Having performed this and possibly other substitutions and alias expansions, the server scans the components of the resulting path. For example:

```
/usr/local/www/cgi-bin/admin/Update-w-Taker.pl/mysite/dev/
```

The HTTP server notices that /usr, /usr/local, . . . /usr/local/www/cgi-bin/admin are all directories, whereas /usr/local/www/cgi-bin/admin/Update-w-Taker.pl is an executable file, residing in a directory that the server knows has an ExecCGI permission. The server checks access restrictions that apply to the script or the /usr/local/www/cgi-bin/admin directory. For example, the server verifies that the client passed hostname/address filtering rules, the user authenticated itself, and the site or directory configuration allows the PUT method. Finally, the server launches the Update-w-Taker.pl script, passing the payload data from the request body to the script's standard input. The request headers and the client and server identification are passed via the process environment:

```
CONTENT_LENGTH=10
CONTENT_TYPE=text/plain; filename="data.txt"
DOCUMENT_ROOT=/w/data/htdocs
GATEWAY_INTERFACE=CGI/1.1
HTTP_HOST=localhost:80
HTTP_USER_AGENT=UPTOW/1.3
PATH_INFO=/mysite/dev/
PATH_TRANSLATED=/w/data/htdocs/mysite/dev/
QUERY_STRING=
REMOTE_ADDR=127.0.0.1
REMOTE_PORT=34022
REQUEST_METHOD=PUT
REQUEST_URI=/cgi-bin/admin/Update-w-Taker.pl/mysite/dev/
SCRIPT_NAME=/cgi-bin/admin/Update-w-Taker.pl
SERVER_ADMIN=oleg@hostname.org
SERVER_NAME=hostname.org
SERVER_PORT=80
SERVER_PROTOCOL=HTTP/1.0
SERVER_SOFTWARE=Apache/1.3.6 (Unix)
TZ=GMT
```

In particular, REQUEST_METHOD tells the method: PUT in our case. All but the well-known request headers are passed as environment variables whose names start with "HTTP_", e.g., HTTP_HOST and HTTP_USER_AGENT. If we submitted a request with the header X-Filename, the CGI script would check for an environment variable HTTP_X_FILENAME. When the request method is PUT, CONTENT_TYPE and CONTENT_LENGTH environment variables must be present to tell the message size and data format.

When parsing the transformed URI above, the server stopped at /usr/local/www/cgi-bin/admin/Update-w-Taker.pl. But the URI continues with /mysite/dev/. This string, if not empty, becomes the content of the environment variable PATH_INFO. The HTTP daemon treats this information as a string – the server does not make any attempt to check whether this string represents a local file, or even whether the string is a valid path string at all.

The content-type of a submitted file must be either

```
application/x-octet-stream-b2a; filename="data.txt"
```

or

```
text/plain; filename="data.txt"
```

This content is stored in a file with the given "basename" in a directory specified by the PATH_INFO parameter, after prepending the $Dest_root. Thus it is generally impossible to place the content outside the $Dest_root tree. Alas, symbolic directory links may defeat this safeguard. The target file is created if needed. The script must have permissions to write into this file or create it. This script responds in a "201 Created" message or in one of the HTTP error codes. All Taker's activity is logged.

Note that both uptow and Update-w-Taker.pl scripts are (deliberately) written using only the most basic facilities: the core Perl and a Socket module. Therefore it is trivial to rewrite the script in some other language, such as Python or TCL.

Note that both uptow and Update-w-Taker.pl scripts are (deliberately) written using only the most basic facilities: the core Perl and a Socket module. Therefore it is trivial to rewrite the script in some other language, such as Python or TCL.

HTTP Versus FTP as a File-uploading Protocol

HTTP is a stateless protocol requiring only a single TCP connection, and therefore less resource-hungry than ftp.

Both ftp and HTTP can be used to upload files from within a firewall. HTTP, however, is designed to operate transparently through proxies and gateways, while ftp requires special SOCKS, etc.–enabled clients and possibly a PASSV mode.

HTTPFS can rely on authentication mechanisms already built into Web servers, in addition to its own access control.

Whenever a file gets uploaded, a receiving HTTP server can synchronously fire up triggers and run arbitrary hooks. This is very difficult to accomplish with ftp. Moreover, if an uploaded file is meant to be fed into an application (e.g., tar, content indexer, META-tag creator, etc.), a receiving HTTP server can launch an application and have it process data *while* it arrives. There is no need to save incoming data to a file and then pass it to an application. HTTP offers similar advantages over ftp as a file downloading tool.

HTTP and ftp also differ in how tightly they couple a client and a server. When an ftp client uploads a file, it has to perform a cd and possibly chmod, ren, and other operations on a remote server, in addition to the PUT operation. If an administrator of the remote site wishes to have the content put under a different name in a different location, she cannot do that unless she talks to the user making an upload and gets him to change the cd command. During an ftp session a client exercises control – albeit limited – over a server. This is not the case with an HTTP upload. A client does not perform any directory navigation or file operations on the server site. The client merely hands over the data and indicates desired file and directory names and similar meta-information. It's up to the receiving server to store, process, or even discard the content as the server thinks fit. The client has no idea of or control over the way the server processes the submitted data. That means a server administrator can change handling of the incoming content at will – and the client will never know or care.

REFERENCES

[1] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1" *<http://www.w3.org/Protocols/rfc2616/rfc2616.html>*

[2] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, E. Sink, and L. Stewart, HTTP Authentication: Basic and Digest Access Authentication, RFC 2617, June 1999. *<http://www.ietf.org/rfc/rfc2617.txt>*

[3] tcp-transactor— a shell tool. *<http://pobox.com/~oleg/ftp/Communications.html#tcp-trans>*

Advanced Applications of HTTP

HTTP can be used for far more advanced tasks – for example, to support network filesystems like NFS or the Andrew File System (AFS). Moreover, HTTP can trivially implement the "Semantic File System" by Gifford et al. That filesystem builds virtual directories wherein the name of a directory corresponds to a query, and the content of the directory consists of files that match the query (represented by symbolic links in the original implementation). Indeed, HTTP provides a file-centric access to remote resources, which can be anything that a server knows how to apply GET/PUT/DELETE methods to. To a client, the exact nature of reading and writing of the resource is irrelevant – to the client, they all look like files.

A particular HTTP-based network virtual filesystem is described at *<http://pobox.com/~oleg/ftp/HTTP-VFS.html>*. It allows one to access, create, and modify remote files as if they were on a local filesystem and to handle RFC822 email messages as if they were local read-only directories. Each email header with the message's body constitutes a "file." An advantage of HTTPFS is that it lets one develop XML, etc., "filesystems" quickly, without any need to modify the kernel.

Appendix: HTTP Uploading Tool

```
#!/usr/local/bin/perl -w
#
# This is a script to publish a file on a remote web site via HTTP
#
# This script is a client part of a HTTP copy facility, with Update-w-Taker.pl
# CGI script being a server part. Both scripts can be downloaded from
# http://pobox.com/~oleg/ftp/Perl
# See also http://zowie.metnet.navy.mil/~spawar/JMV-TNG/Publishing.html
# for more details. The client-server system this script is a part of
# is rather similar to FrontPage's server extensions.
#
# Synopsis:
#    uptow dest-directory local-filename
#
# This script will copy the file specified by the 'local-filename' to a
# remote site. It will be placed into a given 'dest-directory'
# on the remote site under the same (base) name. The remote site will
# typically prepend a pre-defined path to this 'dest-directory'
# (e.g., /usr/local/htdocs or /w/data) to confine file updates
# to that part of its filesystem.
#
# $Id: uptow.pl,v 2.0 1999/11/02 20:58:49 oleg Exp oleg $

     # Configuration parameters
$PROXY_NAME =""; # if empty, no proxy is used
$PROXY_PORT = 80;
$REMOTE_HOST = "localhost";
$AUTH_CREDENTIAL = "Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ=="; # if empty, it is not used
$REMOTE_PORT = 80;
$TAKER_URI = "/cgi-bin/admin/Update-w-Taker.pl";
$USER_AGENT = "UPTOW/1.3";
$CRLF="\r\n";
#$TAKER_URI="/cgi-bin/oleg/test-cgi-my";
use integer;
use Socket;

my $buffer;                # i/o (socket) buffer...
my $transfer_chunk = 1024;
```

```
    # Main module
@ARGV == 2 or &help("Two arguments are expected");
my $dest_dir = $ARGV[0];
my $file_name = $ARGV[1];
$file_name =~ m!([^\\/]+)$! or die "Invalid filename $file_name";
my $base_name = $1;

    # Check the file to publish...
stat($file_name);
-r _ || die "The file to publish – $file_name – does not exist, or unreadable";
my $file_size = -s _;
open(FILE_CONTENT,$file_name) || die "Failed to open $file_name: $!";
binmode FILE_CONTENT;
my $encoding = $ENV{windir} || -B FILE_CONTENT; # On WinNT, -B is not implemented
$encoding && print STDERR "File $file_name appears to be binary and will be encoded\n";

print STDERR "Sending $file_name of $file_size bytes...\n";
my $resource_to_put = "$TAKER_URI/$dest_dir/";
$resource_to_put =~ s![/\\]+!/!g;  # replace double-slashes-backslashes with a single slash

    # Establish the connection with a server
$|=1;             # Set autoflush on...
my $host_to_connect = $PROXY_NAME || $REMOTE_HOST;
my $iaddr_to_connect = inet_aton $host_to_connect;
$iaddr_to_connect || die "Can't resolve the remote host or proxy name $host_to_connect: $!";
my $port_to_connect = $PROXY_NAME ? $PROXY_PORT : $REMOTE_PORT;
print STDERR "Connecting to $host_to_connect:$port_to_connect...\n";
socket(SOCK, PF_INET, SOCK_STREAM, getprotobyname('tcp')) || die "socket: $!";
connect(SOCK, sockaddr_in($port_to_connect, $iaddr_to_connect)) || die "Failed to connect: $!";
binmode SOCK;
print STDERR "Connection established!\n";

    # Making the request (first in $buffer)
$buffer = "PUT " .
    ( $PROXY_NAME ? "http://$REMOTE_HOST:$REMOTE_PORT" : "" ) .
    $resource_to_put . " HTTP/1.0" . $CRLF;
$buffer .= "Host: $REMOTE_HOST:$REMOTE_PORT" . $CRLF;
$buffer .= "User-Agent: $USER_AGENT" . $CRLF;
$AUTH_CREDENTIAL and $buffer .= "Authorization: $AUTH_CREDENTIAL" . $CRLF;
$buffer .= "Content-type: " .
    ( $encoding ? "application/x-octet-stream-b2a" : "text/plain" ) .
    '; filename="' . $base_name . '"' . $CRLF;
$buffer .= "Content-Length: " .
    ( $encoding ? $file_size + $file_size : $file_size ) . $CRLF;
$buffer .= $CRLF; # End-of-headers

syswrite SOCK,$buffer,length($buffer) || die "Request sending error: $!";

my $to_read = $file_size; my $res;
if( $encoding )
{
while ( $to_read > 0 &&
        ($res = read FILE_CONTENT,$buffer,
        ($transfer_chunk < $to_read ? $transfer_chunk : $to_read))) {
    syswrite SOCK,unpack("H*",$buffer),$res+$res || die "socket write error $!";
    $to_read -= $res;
}
} else {
while ( $to_read > 0 &&
        ($res = read FILE_CONTENT,$buffer,
        ($transfer_chunk < $to_read ? $transfer_chunk : $to_read))) {
    syswrite SOCK,$buffer,$res || die "socket write error $!";
    $to_read -= $res;
}
}
$to_read == 0 || die "Failed to read the input file completely: $!";
close FILE_CONTENT;
print STDERR "Request sent\n";

# Read the status line – the first line of the response...
sysread SOCK,$buffer,$transfer_chunk || die "Error reading the status line: $!";
$buffer =~ m!^HTTP/1.\d+\s+(\d+)\s+(.+)!|| die "Invalid status line: $buffer";
my $response_code = $1;
print STDERR "Status: $response_code $2\n";

# Read the rest of the response and dump it...
print $';
while( ($res = sysread SOCK,$buffer,$transfer_chunk) > 0 )
{
print $buffer
}
close SOCK;

if( $response_code == 304 )
{
print STDERR "Not Modified\n";
exit 1;
}

if( $response_code >= 300 )
{
print STDERR "Error\n";
exit 4;
}

print STDERR "Success\n";
exit 0;

# Print help as how to use the program. Print $1 as the title
sub help {
$_ = shift;
print STDERR "\n$_\n";

open(THIS_SCRIPT,"$0") || die "Can't open this script to print out help, due to $!";
while( <THIS_SCRIPT> ) {
/^\#!/ && next;
/^\#/ || last;
print STDERR $'
}
close THIS_SCRIPT;
exit 4
}
```

electronic privacy in the workplace

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation, and other technology issues.

<John.Nicholson@ShawPittman.com>

We spend roughly a third to a half of our waking hours at work.[1] Many of our friends are our coworkers. We communicate with our friends, schedule events, and plan our lives while at work using the work phone, voicemail, email, and pagers and cell phones provided by our company. We generally use these tools as if they were private and secure. As we all know, however, phone calls can be monitored and recorded, voicemail and email can be retrieved, things like IRC or other messaging tools can be recorded, keystrokes can be logged, and so on. To what extent does an employer have the right to monitor employees, and what rights to privacy do those employees have?

The short answer is, the company owns the computer and the company owns the network – if you want something to be private, don't do it at work.

There are several reasons why a company needs to be able to have access to an employee's electronic files and to monitor email and Internet traffic.

First, there is the simple business need of a company to be able to access an employee's files if the employee is not available or leaves the company. Theoretically, email sent to an employee and files stored on an employee's computer should be for business purposes, and, therefore, property of the company, and the employee is only using them on behalf of the company as part of the employee's job.

Second, excessive Internet access and Web surfing can have a negative impact on employee productivity.

Third, companies need to accurately evaluate their bandwidth needs, and to do so they need to be able to separate legitimate business-related use from employee surfing and downloading.

Finally, in the current era of increasing employer liability for the activities of employees, employers have little choice but to monitor employees' actions, including their use of employer-provided electronic tools. Prior to the advent of email and the Internet, employers and employees did not have as much to worry about. Now, however, the ease with which large quantities of information can be sent to multiple people creates a situation ripe for disgruntled employees to divulge company secrets and for workers to be offended by their coworkers' sense of humor. Additionally, the casual and spontaneous nature of email may allow employees to write things that would not look good when presented to a jury. Moreover, the seeming privacy and anonymity of email and the Internet makes some people do or say things they would not do or say if they thought they might be seen or overheard by a third party. Employees who email sexually explicit or racially based jokes or who download pornographic or explicit images can offend coworkers and, according to the courts, can create a "hostile workplace."[2] Unless employers can show that they have policies in place that prohibit such use and that they take action against those who violate such policies, employers can be held liable for substantial damages and be subject to a lot of very bad press. To protect themselves, employers have a very real need to be able to monitor employees' electronic activities and to search employees' computers for specific files. The question is, how far can employers go in monitoring and searching employees' electronic activities?

As I discussed in my last column, the first place to look when asking a question like this is whether there is a federal statute on the subject. In this case, Title 18 of the Omnibus

While at work or while connected to your company's network, behave as if everything you read or type is also being read and saved by someone in the IT department.

Control and Safe Streets Act (the "Safe Streets Act") prohibits all private individuals and organizations, including employers, from intercepting the wire, oral, or electronic communications of others.[3] Generally, the Safe Streets Act provides for penalties and civil damages when a third party intercepts a telephone conversation without the consent of either of the conversing parties.[4] In 1986, the Electronic Communications Privacy Act (more commonly known as the Federal Wiretap Act) amended the Safe Streets Act to cover interception of electronic communications.

The Safe Streets Act now prohibits the intentional interception of any "wire, oral, or electronic communication," and defines "intercept" as the "aural or other acquisition of the contents of any wire, electronic, or oral communication through the use of any electronic, mechanical, or other device."[5] Under the Safe Streets Act, intercepting an electronic communication "means acquiring the transfer of data."[6]

There is, however, a significant exception to this provision. Section 2701(c)(1) of the Safe Streets Act allows the provider of an electronic communications service to do virtually whatever it wants with regard to accessing electronic communications.[7] Thus, if the communications service is being provided by the employer, the employer has virtually free access to the communications. However, if the communications service is being provided by a third party, and any messages are stored by that third party and not on any service provided by the employer, then the employer does not have a right to access those stored messages. The practical implications of this are:

(1) If you are using a computer provided by your company, your company probably has a right to retrieve stored files from your computer (including personal email, graphics files, Web-browser caches, logs, etc.).

(2) If you are sending email or using some messaging service inside your company's network, the company can probably read and save any email or messages you send or receive, regardless of whether you are using a company-provided computer or not (note that if you are also using a company-provided computer on the company's network, (1) will also apply).

(3) If you are using your company's network to access the Internet, and you are using a third-party email service, then your company can monitor the traffic that goes in and out of its network (including recording the Web sites that you visit and reading and saving email or messages that you send or receive), but the company cannot access any email or files saved on the third party's system without the approval of that third party or the sender or receiver of the message.

Thus, the proper way to handle your computer while at work or while connected to your company's network is to behave as if everything you read or type is also being read and saved by someone in the IT department.

Careful readers will note that I used the word "probably" in the above list. In general, a federal law will take precedence over a state law, especially if the monitored communications are interstate communications. However, it is possible that there could be situations where some state's constitution or laws could limit the type and scope of employee monitoring in the workplace. Alaska, Arizona, California, Florida, Hawaii, Illinois, Louisiana, Massachusetts, Montana, Rhode Island, South Carolina, Washington, and Wisconsin all have laws or provisions in their state constitutions regarding rights to privacy.[8]

If your company operates in multiple states, or if your company has operations outside the U.S., be sure to have your general counsel review the laws of those states or countries and the policies of your company to determine what you are allowed to do and what privacy rights your company's employees in those states or countries may have. Additionally, whether a company has a usage policy that states that electronic communications should be used exclusively for business purpose and can be monitored and recorded, and whether an employee has explicitly signed a statement acknowledging that the employee has read and consents to the policy, can affect whether a court determines that a company's monitoring of an employee or search of that employee's electronic files was a violation of the employee's right to privacy.

For an example of how a U.S. federal court has approached these issues, let's look at the case of U.S. v. Simons.[9] In that case, a network manager was checking his firewall logs and noticed that the logs were unusually large. Upon scanning the logs, he noticed that several of the requests were for Internet Web sites, including one that appeared to be pornographic and, therefore, not for legitimate business purposes.[10] The network manager also noticed that a significant group of the hits came from a single workstation (in Simons's office). He reported the discovery, and his supervisors first confirmed that the Web site in question was pornographic and then discovered that Simons's workstation had over a thousand pictures stored on it, some of which appeared to be pornographic.

At that point, the supervisors remotely copied the workstation's hard drive. Because some of the pictures appeared to be child pornography, Simons' employer notified the FBI. The FBI obtained a search warrant for Simons's office and made a copy of his workstation's hard drive, disks that were found in his desk, and documents relating to screen names and other personal correspondence.

In analyzing whether the search of Simons's office was legal, the court focused on whether Simons had a reasonable expectation of privacy.[11] The court found that Simons's employer had an official policy regarding Internet use that stated, "permitted use includes official business use, incidental use, lawful use, and contractor communications." Simons's downloading of child pornography did not fall into any of these categories. Additionally, Simons's employer had a policy regarding audits that stated:

> Audits. Electronic auditing shall be implemented with all . . . networks that connect to the Internet or other publicly accessible networks to support identification, termination and prosecution of unauthorized activity. These electronic audit mechanisms shall . . . be capable of recording:
>
> - Access to the system, including successful and failed login attempts, and logouts;
> - Inbound and Outbound file transfers;
> - Terminal connections (telnet) to and from external systems;
> - Sent and received e-mail messages;
> - Web sites visited including uniform resource locator (URL) of pages retrieved;
> - Date, Time and user associated with each event.[12]

Because of this policy, the court did not find that Simons had a reasonable expectation of privacy with regard to any Internet use. Thus, his employer's search and copying of his workstation's hard drive was not an unreasonable invasion of Simons's privacy.

The court did not find that Simons had a reasonable expectation of privacy with regard to any Internet use. Thus, his employer's search and copying of his workstation's hard drive was not an unreasonable invasion of Simons's privacy.

Conclusion

The most important thing for employers to do regarding employee privacy is to be open and clear about the company's intentions. If the company intends to monitor employee activities and/or communications, that needs to be made clear in company policies, training, and demonstrated actions. A company should develop a policy on privacy that reduces its employees' expectations of privacy. The lower the employees' expectation of privacy, the better the company's chances in any future litigation challenging the propriety of any monitoring or search.

A company should do everything it reasonably can to minimize its employees' expectations of privacy – consistent with the company's culture and without damaging employee morale. At the same time, it would not be overly paranoid to suggest that employees should assume that every electronic communication could be read or heard (and read or heard out of context) by a third party, and could be used as evidence in a lawsuit against the company or as grounds for termination of the employee.

NOTES

[1] This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

[2] A hostile workplace is a term for employment discrimination consisting of unwelcome verbal or physical conduct (as comments, jokes, or acts) relating to the victim's constitutionally or statutorily protected classification (as race, religion, ethnic origin, or age) that has the effect of substantially interfering with a person's work performance or of creating a hostile work environment.

[3] 18 U.S.C. §§ 2510-2520.

[4] 18 U.S.C. § 2511(1)(4)(a).

[5] 18 U.S.C. § 2511(1)(a); § 2510(4).

[6] United States v. Reyes, 922 F. Supp. 818, 836 (S.D.N.Y. 1996).

[7] 18 U.S.C. §2701(c)(1).

[8] Alaska – Const. Art. I, §22; Arizona – Const. Art. II, §8; California – Const. Art. I, §1, Lab. Code §2930; Florida – Const. Art. I, §23; Hawaii – Const. Art. I, §6; Illinois – Const. Art. 1, §6; Louisiana – Const. Art. 1, §5; Massachusetts – ch. 214, §1B; Montana – Const. Art. II, §10; Rhode Island – §9-1-28.1; South Carolina – Const. Art. I, §10, §30-4-50; Washington – Const. Art. I, §7; Wisconsin – §895.50.

[9] 29 F.Supp.2d 324 (E.D. Va. 1998).

[10] In this case, the subtle and well-disguised *<http://www.xratedpictures.com>*.

[11] Ibid. at 326. ("The person must have had an actual or subjective expectation of privacy and the expectation must have been one that society recognizes as reasonable.")

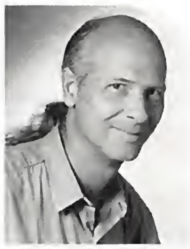
[12] Ibid. at 327.

musings

Ah, if life were only that simple!

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*



<rik@spirit.com>

The world we live in is a dangerous place, and it just got a bit more unpleasant. Of course, our daily travails are nothing compared to those of people living on those islands in Indonesia where the Christians and Moslems take turns killing one another, or in the Russian province of Chechnya, which perhaps will have settled back into relative calm by the time you read this.

But even in the Western world, there are dangers to face. For example, I set out on a journey recently. The first phase of that journey entailed hurtling through the desert at 110 feet per second (about 33.5 meters per second). Then I entered an area known to be dangerous. Armed guards were stationed at strategic locations, and abandoned vehicles were quickly towed away lest they explode unexpectedly. Everyone entering the facility was searched for guns and explosives, and there were frequent announcements that any "unusual or suspicious" behavior should be reported immediately.

I am talking about going to the airport, of course. Many of us take things like driving on crowded freeways and traveling in airplanes (where a single defective part might send us diving to our deaths) as part of everyday living. I admit this is a great advance over the "good old days," when living to 45 was considered a great feat, and any travel at all took many times longer, with a much greater risk. Still, it is no wonder that we all feel a little stress at times.

And just this week as I write this, the second week in February, something long dreaded by many of us in the security industry has come to pass. As if ordinary network-based denial-of-service (DoS) attacks were not enough, distributed DoS attacks using hundreds of agents have begun taking down e-commerce sites.

Not New

For some people, this is not news at all. Distributed DoS attacks were the topic of this year's first CERT advisory, and this advisory actually preceded widespread attacks. But there had been attacks. Attackers targeted an IRC server at the University of Minnesota last summer, disabling not only that server but also most of the university's network infrastructure for two days. During this attack, over 2,000 different remote systems participated in sending floods of UDP packets. The University of Minnesota has an OC3 connection to the Internet, and another OC3 to Internet2, and both of these links remained swamped for days.

The only "bright side" to this attack is that the tool used, Trinoo, does not spoof source addresses, making it relatively easy to find the agents sending the floods. But it was the scale of the attack that was mind-boggling. John Ladwig, security architect, Networking and Telecommunications Services, wrote me saying that "it frightened me that someone would 'throw away' approximately 2,000 compromised hosts, primarily very well-connected [and] fairly powerful ones, presumably to seize IRC channels." This happened in August of 1999, and by February 2000 the attacks had a new level of visibility, even gaining the attention of Attorney General Janet Reno and spots on all the major news sources. Reno promised to prosecute the offenders, something I found interesting. Current US law does not include denial-of-service attacks, but covers only "unauthorized use." Since these attacks come from systems that have been broken into, the perpetrators can be prosecuted because they broke into systems, but not for the denial of service.

There are other attack tools – Tribal Floodnet, TFN2K, and Stacheldraht – all of which can spoof source addresses. If you are the victim of one of these attacks, it may not be immediately obvious that addresses are spoofed, because they will be coming from

Just imagine what fun you would have trying to convince a Russian or an Indonesian network administrator that some system at her site has launched a devastating attack on you.

many different sources. Even if the addresses are not spoofed, you will need the cooperation of many remote sites to stop the attacks. In many cases, ISPs' systems were used as agents, as these systems are not only usually powerful UNIX systems, but by definition are connected to relatively fast network links. Another "bright side" to the recent attacks is that most of them come from the continental United States, against other U.S. targets, so the owners of the attack daemons and the victims both live in similar time zones and speak English. Just imagine what fun you would have trying to convince a Russian or an Indonesian network administrator that some system at her site has launched a devastating attack on you. (Actually, if you have ever experienced handling situations like this, please contact me.)

Dave Dittrich of the University of Washington wrote several very useful analyses of these attack tools that you can download from his Web site: <*http://staff.washington.edu/dittrich/misc/[trinoo.analysis|tfn.analysis|stacheldraht.analysis]*>. If the tools have not been modified, you can use Perl scripts written by Dave to search for daemons within your own networks. Some of the daemons are programmed to respond to certain requests, and his Perl scripts trigger these responses – again, only if the daemons have not been modified so that the requests are now invalid. Also, if you are interested, you can download the attack tools themselves (unless the Feds have blocked this) from <*http://www.technotronic.com/denial.html*>. David Brumley, who works on security at Stanford, has written a tool for detecting daemons that will compile on Solaris and Linux: see <*http://www.theorygroup.com/Software/RID*>.

The various intrusion-detection vendors have added capabilities to detect master-daemon communications, and some scanners include the ability to send the triggering requests that will flush out daemons hidden in your networks.

The terminology I am using comes from the CERT advisory, <*http://www.cert.org/advisories/CA-99-17-denial-of-service-tools.html*>. Masters are the systems used to control the daemons, the software that actually does the flooding. The attacker connects to the master – sometimes using a special client, other times just by telneting to a port and entering a password – and uses the master to send commands to the daemons. TFN2K daemons run as root and sniff the network, looking for the encrypted commands that come from masters. No responses are sent back to the masters for routine commands, so there is no two-way communication that can be sniffed and used to discover masters. TFN, TFN2K, and Stacheldraht can all use ICMP packets for communication.

Masters and daemons sometimes have lists of addresses of their counterparts either built in or located on the local system, for example, in a file named ... [three dots]. If you find masters or daemons, don't just delete them! They may contain information leading to the discovery of still more exploited systems. Again, Dave Dittrich's papers have information on recovering lists of systems (which are sometimes encrypted, but with a key that can be recovered from the executable). If you do not wish to deal with this yourself, try CERT.

Are You Vulnerable?

On a barely related topic, the Chinese launched their own attack against U.S. government sites after the accidental cruise-missile bombing of their embassy in Belgrade in the fall of 1999. The Chinese attacks revealed that they had previously compromised over 4,000 systems, installing backdoors for later access and use.

I've mentioned over 6,000 compromised systems so far in this article, an infamous number. There were 6,000 systems exploited by the Internet Worm in November of

1988, at that time a large proportion of Internet servers and gateways. Today, 6,000 systems is barely a handful of the servers connected to the Internet. But the real question is, "Why are all these systems vulnerable?"

Why are there thousands of systems waiting to be compromised? Usually because existing security patches have not been installed.

Distributed DoS attacks would not succeed without a ready supply of exploitable systems. Dave Dittrich's papers, and conversations with David Brumley of Stanford Computer Security, reveal the manner in which these systems are compromised. The attacker uses scanning tools that search through large ranges of IP addresses looking for vulnerable systems with tools like `mscan` and `sscan`. Then the attackers break into one of these systems and launch a script that uses the same exploit to break into a list of other systems. The exploit involves using `rcp` to the script-running system to download the agents and launch them. Brumley wrote "Often we'll find a list of hundreds of compromised hosts (usually because an intruder is rcp'ing over a rootkit and the rcp's are logged in SYSLOG) on another site that the administrator was ready to just delete!" More reasons to look before you wipe that compromised system clean.

And why are there thousands of systems waiting to be compromised? Usually because existing security patches have not been installed. I wrote a "passionate plea" in my last column about choosing one or two good ways of accomplishing this and other tasks, instead of inventing new ways for each LISA conference.

Another reason has to do with the lack of firewalls. Many sites avoid firewalls because they have an open policy. If all the firewall did was enforce RFC2267, that would help a great deal. RFC2267 describes ingress and egress filtering, techniques for blocking packets that have spoofed source addresses. Unless your site is a transit site – that is, you carry traffic for all parts of the Internet – you know exactly what source addresses should appear in packets leaving your site. By blocking packets that do not have one of those source addresses, you can prevent source address spoofing from your own site.

RFC2267 was written in response to the spate of SYN flooding that occurred after people unraveled the attack on Tsutomu Shimomura back on Christmas Day, 1996. SYN flooding became very fashionable among small ISPs that were attempting to damage competitors' reputations by flooding their Web sites and filling the TCP connection queues. RFC2267 is even more relevant today, with the increasing popularity of DoS attacks, where spoofing the source address makes sense and is easy to do.

Firewalls could also be used to block the widespread scanning that takes place. Or perhaps just to block the "r" commands (or does your site routinely trust everyone on the Internet, as was done in the old default configuration of Sun workstations)?

Force

If the recent attacks have continued, perhaps we will soon be forced to impose RFC2267 on our networks. In my opinion this would be a good thing, and it might come about in any of several ways. One way would be for the Network Access Providers (NAPs) to deny connectivity to sites that do not apply RFC2267 filtering. While tedious to set up (for larger sites), and requiring still faster and more expensive routers (or firewalls), this is nothing more than a good neighbor policy. That is, we will be good neighbors and not pollute the Internet with spoofed source packets from miscreants within our own networks. University of Minnesota does RFC2267 filtering all the way out to the edges of its networks, which encompass 50,000 systems. Stanford University filters at its borders.

Another way this might come about is through legislative action, but I am not holding my breath for that one. Perhaps the most likely, and the least pleasant, will be the civil lawsuit. eBay will sue one or more large universities or organizations as being the source of the floods against their Web servers that cost them hundreds of thousands (millions?) of dollars in possible revenues. If a suit like this were successful, or just long-lasting and unpleasant enough, network administrators would suddenly find themselves with the resources to make their networks more secure. That is, due diligence would be required, or the organization would face a lawsuit.

The icing on this cake came when CERT put out its second advisory of 2000, warning that it is possible for evil sites to include information in links that can cause scripts to be executed on clients visiting trusted sites. The advisory (*<http://www.cert.org/advisories/CA-2000-02.html>*) suggests that Web-site operators revise any scripts that create forms to filter out HTML that may have been added by a potential attacker. This advisory also suggests disabling all forms of active scripting (Java, JavaScript, and, by implication, ActiveX started by JavaScript), something that many of us have already done. Sorry, Microsoft, but you will have to come up with another way of world domination. Perhaps giving away Windows 2000? In late February, a version of the trinoo attack tool appeared as a trojan horse installed along with Back Orifice on Windows systems, so UNIX systems are no longer the only host for attack daemons for DDoS.

Sigh. The world is a dangerous place. Still, I am glad that my neighbors keep their AK47s unloaded and put away, and that no one is launching artillery or rocket attacks against my neighborhood.

performance tuning with source code UNIX

This issue's column will focus on performance improvement – an activity requiring technique, intuition, and of course source code. Most code bases have room for large improvements - but an analysis is required to determine if the payoffs are worthwhile. We'll be talking about the intuition for code efficiency that takes years to develop, and we'll examine the engineering process that leads to faster code. Finally, we'll look at some code-tuning success stories.

by Bob Gray

Bob Gray is co-founder of Boulder Labs, a software consulting company. Designing architectures for performance has been his focus ever since he built an image processor system on UNIX in the late 1970s. He has a Ph.D. in computer science from the University of Colorado.

<bob@cs.colorado.edu>

For 20 years, CPU speeds have doubled every 1.5 years. The VAX 11/780 that was widely available in 1980 ran at one SPECMARK.[1]

We now bask in the land of cheap CPU cycles with affordable machines running almost one thousand times faster. Then why do we frequently hear complaints about "sluggish" programs or "slow- booting" operating systems? The layperson might think that a 400MHz Pentium should "feel" three times faster than a 133MHz Pentium, but the new one still takes 60 seconds to boot. And why isn't opening documents and receiving mail three times faster?

We can look at three root causes for slowness complaints:

1. The task is computationally intense and just takes a long time – for example, image processing, multimedia encoding/decoding, and certain numerical problems. Here we will notice that faster CPUs make a big difference.
2. The slowness is due to I/O waiting, and faster CPUs won't help. Web-page loading over a 28Kbps connection will require about the same elapsed time on a Pentium-400 as on a Pentium-133. Similarly, the time required to search for something in the file system is largely independent of the CPU speed.
3. The code being run is suboptimal, and there is room for improvement. Here is where we can make a difference by tuning the application or adjusting its data structures and algorithms.

Performance Intuition

Have you ever run into a programmer who says, "That code should be able to run much faster," without ever having seen the code base? How did he or she know there was room for improvement? Over the years, one acquires a feel for what it takes to get a job done. Jon Bentley has written an outstanding book, *Programming Pearls* (2nd ed., ACM Press, 1999) to help a programmer develop this intuition. He poses numerous problems and goes through the thinking process required for an efficient solution. Bentley talks about back-of-the-envelope calculations to perform sanity checks on system designs. For example, what size computer system would be required to store, index, and retrieve ten years' worth of newspaper articles? Would a laptop be sufficient, or would you want a multiprocessor enterprise server? Given that we store 365 x 10 days, we only need to estimate how much storage is required per day to perform a sanity check on the server size. You'll employ this intuition when your client says "performance needs to be improved." Now is the time to ask lots of questions and get the client to commit to specific assumptions.

Until you can quantify your program's behavior in terms of runtime, you cannot systematically attack it. To establish the baseline, you'll need both performance monitors and a repeatable input sequence.

Methodology for Code Improvement

How do you go about making improvements to a code base? You'll be faced with numerous constraints and goals, and your first task will be to find the acceptable subset of the solution space. How much time is available for the project? How much effort can be expended? Do you have the right kinds of people to work on the project? What size improvement is being sought? Does it need to run 20% faster or 20 times faster? Can you spend money on equipment and solve the problem with hardware? Once some guidelines are established, you will be able to home in on the appropriate changes.

Before changing the code base you should first establish "baseline" performance measurements. Until you can quantify your program's behavior in terms of runtime, you cannot systematically attack it. To establish the baseline, you'll need both performance monitors and a repeatable input sequence. The simplest monitor is a stopwatch, but you may need finer-grained timers. UNIX systems provide microsecond timers (for example, getrusage) that can identify process "user" time, "system" time, and elapsed time. Occasionally, you may need access to kernel nanosecond timers – they are available on most workstations and recent PCs.

You should find a way to get repeatable time measurements for a certain task. You may need to isolate the test machine so that other people's work won't interfere with your measurements. You'll also want to run the same input data or the same sequence of mouse clicks. Be aware of the "warm cache" effects – where successive runs of a program require much less time because the system has already "seen" your code and data. Performance tuning will require the following steps:

1. Form a hypothesis of what is happening in the system.
2. Establish baseline performance measurement.
3. Analyze what is happening in the system.
4. Implement code change.
5. Measure the effects of the changes.
6. Repeat steps as necessary.

Most skillful code tuners are not geniuses – they just methodically attack the problem and carefully think about what is happening. Solutions then are often straightforward.

Tools

Profilers are tools for measuring the execution times of code fragments. Most systems provide the typical subroutine profiler. It measures the time spent executing subroutines, either by sampling the program counter or by collecting data by adding subroutine prologues and epilogues.

Finer-grained profilers such as tcov measure the time spent executing blocks of code or even individual lines of code. Sometimes these tools are well suited for numerical analysts that must pay attention to the inner loops of calculations.

When I need to understand the performance characteristics of a body of code, I resort to a call-graph profiler. I use gprof, which attributes times to program abstractions. For example, we may have a collection of subroutines that perform a logical task. Even though none of the task's individual routines shows up as "expensive," the whole body of code taken together may show that the runtime of the program is dominated by the one logical task. gprof highlights this information and shows the programmer what abstractions should be scrutinized or replaced.

Table 1 shows the twentieth [20] most expensive abstraction is rxl_PrintBuff. It is called 100 times from xp_emulate and 579 times from zp_emulate. The abstraction that

index	% time	self	children	called	name
		0.02	1.14	100/679	xp_emulate [57]
		0.02	2.14	579/679	zp_emulate [37]
[20]	7.9	0.02	3.14	679	rxl_PrintBuff [20]
		0.01	3.12	1210/1210	rxl_PrintChar [21]
		0.00	0.00	679/817	rxl_SetTexture [471]
		0.01	3.12	1210/1210	rxl_PrintBuff [20]
[21]	7.8	0.01	3.12	1210	rxl_PrintChar [21]
		0.03	2.87	1139/1139	get_char [24]
		0.00	0.21	1089/1089	rxl_FixedCh [92]

Table 1

includes its children consumes 7.9% of the time, or about 3 seconds (0.02 + 3.14). The routine itself isn't expensive, but it calls rxl_PrintChar 1,210 times. Again, rxl_PrintChar itself doesn't consume much time – the child get_char is where the CPU time is spent. The percentage times, by the way, are slightly distorted because of the percentage spent profiling; in reality, they are slightly higher.

Examples

Mike Durian and I were challenged by a laser-printer company to "speed up" printing. Its printer consisted of a RISC processor, a large amount of memory, a disk, and a full-featured source code operating system. We used a number of tools to characterize the printer's behavior, ranging from a stopwatch to various kernel counters. We realized that more sophisticated tools would help zero in on problems and consequentially implemented kernel gprof and a graphical virtual-memory trace facility, as shown in Figure 1. What appears as a pair of parallel horizontal lines is two sets of points representing discrete page-in and page-out events from the time 2700 microseconds to 2760. The upper set of points represents page-out activity that is due to a memory shortage. The lower set shows the corresponding page-in events. Here we have a classic case of thrashing – sending to disk a page that will be needed again

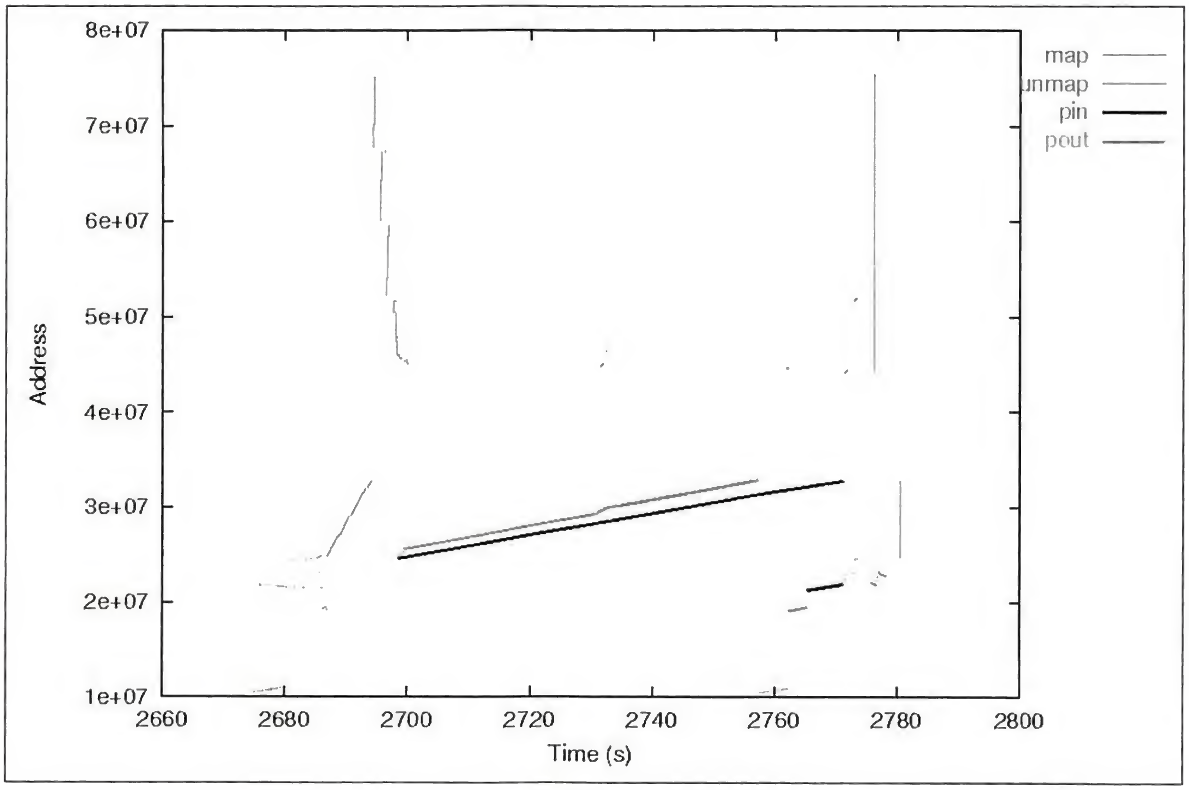


Figure 1

NOTE

[1] SPEC, the Standard Performance Evaluation Corporation, is a nonprofit corporation formed to "establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers." *<http://www.spec.org>*

soon. We implemented a different page-replacement policy, and subsequently it minimized the thrashing.

Information collected from gprof showed a bottleneck in the I/O system. Far too much time was spent waiting for disk blocks during normal system operation. This was true even without thrashing. We implemented a page-clustering algorithm, à la FreeBSD, to amortize the disk seek time over a large chunk of blocks. Our customer had considered upgrading its printers with faster and more expensive disks. Instead, a few hundred lines of code achieved a performance improvement that far exceeded the possible speed gains of quicker disks.

History

In 1980, Bill Joy performed one of the most dramatic examples of kernel tuning. The DEC VAX 11/780 with its "huge" 32-bit address space had just been released. The ARPA community (ARPANET was the predecessor to the Internet) was evaluating which operating system to run on the new hardware. Their choices were: DEC's VMS, UNIX, or a custom-built operating system.

David Kashtan of SRI published some performance numbers that showed VMS to be significantly more efficient in numerous areas. He urged the community to adopt VMS as the operating system base for future ARPA-sponsored code.

A few weeks later, Joy published a memo entitled "Comments on the Performance of UNIX on the VAX." The report details a set of performance improvements that were incorporated into the production systems at Berkeley during the first three weeks of March 1980 (the full paper is available at *<http://boulderlabs.com/joy.html>*).

Joy's paper showed a careful analysis of where the kernel time was spent in UNIX and went through a series of fairly easy modifications that eliminated many of the former inefficiencies. After tuning, he showed UNIX and VMS taking about the same amount of time for most of the benchmarks. While Joy's paper is an elegant piece on system tuning, parts of his conclusion are most insightful:

> In selecting between UNIX and VMS as an operating system for use in ARPA . . . , the UNIX system was chosen primarily out of concern for portability. For our purposes within the Computer Science Department at Berkeley, we have chosen UNIX because of its pliability to meet our needs.
>
> In the short term, there are areas of the UNIX system that are still suffering growing pains from the porting of the system to larger machines. We believe that the simplicity and modularity of the system, which are the keys to its portability, are also the reasons why UNIX is easy to tune, as this paper has demonstrated.

Code tuning is not black magic. Certainly some people are extremely effective in this area; however, anyone willing to apply simple principles systematically can achieve strong results.

java performance

Using the Java Language to Push Bits Around

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

The Java programming language is often cited as a good language for constructing user interfaces and Web applets, multimedia applications, and so on. It has many features that support this style of usage. But how well does the language do at some tasks traditionally handled by C and C++? For example, what if you want to do some low-level bit manipulation? Do you pay a performance penalty?

To partially answer this question, we will look at an example of data compression, one that uses a special type of encoding to compress sequences of small integers. We'll look at the Java source code for this compression algorithm and compare its performance to that of C++.

Rice Coding

Suppose that you're developing an application that constructs indexes for large groups of text files. For each word in the text files, you'd like to create a list (an inverted index) that gives all the file numbers where that word exists. In other words, the word "tree" might be found in files 1, 7, 23, 39, 54, and 79. Each file number represents an actual pathname.

One technique that's been found useful in creating such lists is to store deltas between numbers instead of the numbers themselves, for example:

```
1, 6, 16, 16, 15, 25
```

So the first file number (1) has a delta of 1 from the base (0), the second file number (7) has a delta of 6 from the first, and so on.

The reason deltas are used is that it's possible to compress lists of deltas efficiently, whereas compressing lists of large file numbers is more difficult. Common words in an index will typically have small deltas, and uncommon words often occur in clusters together, with small gaps between the occurrences.

One type of compression that's used in this situation is known as "Rice coding," which is a variation on another technique called "Golomb coding."

Rice coding works as follows: for a positive value *x* to be encoded along with a parameter *m*:

1. Let:

```
b = 2^m
q = floor((x-1)/b)
```

Emit q 1 bits followed by a 0 bit.

2. Emit the lower m bits of x-1, treating x-1 as a binary value.

For example, if m = 2 and x = 7, the coding is:

```
10 10
```

If m = 0 and x = 1, the coding is:

```
0
```

If m = 1 and x = 10, the coding is:

```
11110 1
```

If m = 3 and x = 10, then the coding is:

```
10 001
```

Different values of m are chosen depending on what range of values for x is expected; a typical value for m is a small integer like 5. The value of m can be chosen to minimize the encoded length for a given distribution of numbers. For example, suppose that 95% of the time you expect to encounter numbers in the range 1–25, but 5% of the time the range will be 1–1000. In this case, m = 4 is optimal, and the average number of bits is 6.3 per number. In other words, you can encode 1,000 numbers in about 6,300 bits. The demo program below derives this value empirically.

One simple way of understanding why this coding algorithm works is to view the output values in two parts. The second part emits the bits that make up most of the number, for example, 5 bits if m = 5. The first part of the output is an "escape" or "overflow." If the distribution of numbers is mostly in a narrow range, then this type of coding is very efficient.

Decoding simply reverses the process, to get a sequence of numbers from a stream of bits.

Implementation of the Rice Coding Algorithm

Here is a Java implementation of Rice coding. The driver for the RiceCode class tries out values of m between 0 and 16 and displays the number of bits required to encode 1,000 numbers, distributed such that 95% of the numbers are in the range 1–25 and 5% in the range 1–1000.

This code can be downloaded at <*ftp://ftp.glenmccl.com/pub/free/rice.zip*>.

```
public class RiceCode {
    private static final int BASE = 7;
    private int size;       // number of items stored
    private int nbits;      // number of bits to store the items
    private int m;          // parameter to this encoding
    private byte[] bits;    // the actual bits
    // add a bit to the encoding

    private void addbit(int b) {
        int len = bits.length - BASE;

        // need to grow the bit list

        if (nbits == len * 8) {
            int newlen = (int)(len * 1.5) + BASE + 1;
            byte tmp[] = new byte[newlen];
            System.arraycopy(bits, 0, tmp, 0, bits.length);
            bits = tmp;
        }
        if (b == 1)
            bits[BASE + (nbits >> 3)] |= (1 << (nbits & 0x7));
        nbits++;
    }

    // get the value of the n-th bit

    private int getbit(int n) {
        return (bits[BASE + (n >> 3)] >> (n & 0x7)) & 0x1;
    }
```

```
// construct a new encoding object

public RiceCode(int mval) {
    bits = new byte[BASE];
    m = mval;
    if (m < 0 || m > 16)
        throw new RuntimeException("m < 0 || m > 16");
    bits[BASE - 1] = (byte)m;
}

// construct a new object from a previously
// exported encoding

public RiceCode(byte vec[]) {
    bits = vec;
    int b0 = (bits[0] < 0 ? bits[0] + 0x100 : bits[0]);
    int b1 = (bits[1] < 0 ? bits[1] + 0x100 : bits[1]);
    int b2 = (bits[2] < 0 ? bits[2] + 0x100 : bits[2]);
    int b3 = (bits[3] < 0 ? bits[3] + 0x100 : bits[3]);
    int b4 = (bits[4] < 0 ? bits[4] + 0x100 : bits[4]);
    int b5 = (bits[5] < 0 ? bits[5] + 0x100 : bits[5]);
    size = (b0 << 16) | (b1 << 8) | b2;
    nbits = (b3 << 16) | (b4 << 8) | b5;
    m = bits[BASE - 1];
}

// get the number of items

public int getSize() {
    return size;
}

// get the number of bits

public int getNumBits() {
    return nbits;
}

// add an item to the encoding
public void addItem(int val) {
    if (val < 1)
        throw new IllegalArgumentException("val < 1");
    size++;

    int x = val - 1;
    int q = x >> m;
    int r = x & ((1 << m) - 1);

    // encode the first (unary) part

    while (q-- > 0)
        addbit(1);
    addbit(0);

    // encode the binary part

    if (m > 0) {
        int mask = (1 << (m - 1));
        while (mask != 0) {
            addbit((r & mask) != 0 ? 1 : 0);
            mask >>= 1;
        }
    }
}
```

```
    // get the items in this encoding and return them in a vector

    public int[] getItems() {
        int items[] = new int[size];
        int currbit = 0;
        for (int i = 0; i < size; i++) {
            int unary = 0;
            while (getbit(currbit) != 0) {
                unary++;
                currbit++;
            }
            currbit++;
            int binary = 0;
            for (int j = 1; j <= m; j++)
                binary = (binary << 1) | getbit(currbit++);
            items[i] = (unary << m) + binary + 1;
        }
        return items;
    }

    // export the encoding into a vector of bytes

    public byte[] getBits() {
        bits[0] = (byte)(size >> 16);
        bits[1] = (byte)((size >> 8) & 0xff);
        bits[2] = (byte)(size & 0xff);
        bits[3] = (byte)(nbits >> 16);
        bits[4] = (byte)((nbits >> 8) & 0xff);
        bits[5] = (byte)(nbits & 0xff);
        int len = BASE + (nbits + 7) / 8;
        byte tmp[] = new byte[len];
        System.arraycopy(bits, 0, tmp, 0, len);
        return tmp;
    }

    // driver

    public static void main(String args[]) {
        java.util.Random rn = new java.util.Random(0);
        for (int i = 0; i <= 16; i++) {
            RiceCode rc = new RiceCode(i);

            // add 1000 numbers, with 5% of them in the
            // range 1-1000, 95% in the range 1-25

            for (int j = 1; j <= 1000; j++) {
                int num;
                if (rn.nextInt(20) == 0)
                    num = rn.nextInt(1000) + 1;
                else
                    num = rn.nextInt(25) + 1;
                rc.addItem(num);
            }

            // print out the number of bits used

            System.out.println(i + " " + rc.getNumBits());
        }
    }
}
```

Performance Compared to C++

For timing comparison purposes, this same algorithm was also implemented in C++. The driver program given above was replaced with one that iterates across values of m between 0 and 16 and encodes 4,000 numbers for each:

```
public static void main(String args[]) {
    for (int k = 1; k <= 10; k++) {
        for (int i = 0; i <= 16; i++) {
            RiceCode rc = new RiceCode(i);
            for (int j = 1; j <= 4000; j++)
                rc.addItem(j);
        }
    }
}
```

Using Borland C++ 5.4 as a C++ compiler, and Sun's JDK 1.2.2 with just-in-time compilation for the Java compiler, execution times in seconds on a 300MHz Pentium are like this:

Java	20.5
C++	19.9

The difference here is negligible.

The Java language does not primarily target systems programming, and in fact there are some things that are impossible to do with the language, such as low-level memory manipulation using specific virtual-memory addresses. But for many other programming tasks, these limitations don't matter. We've illustrated one such task.

using java

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi+@ux4.sp.cs.cmu.edu>

The Java platform provides the programmer with a core set of classes that are in "packages," and we have seen many examples (such as java.io and java.net) in previous articles.

The way data is stored and structured in your program can make the difference between a clean, elegant solution and a mass of unrecognizable tangled code. For example, we know that an instance variable of a class can take the place of data structures in other languages; also, an array can hold many copies of one type of data. Nevertheless, we sometimes require "data containers" that are more powerful and flexible than either of these options.

Java provides a set of "collection classes" as part of the java.util package that are ready-made solutions to common data-storage problems. We will examine a few of the collection classes in Java. I'll leave it to the reader to extrapolate the use of the remaining collections by understanding the ones presented here.

Specifically we present the Stack class, Abstract Dictionaries, Property Lists, and Random Numbers.

Collections Explained

Collections in Java are not type-specific and can hold any type of object. This makes a collection a very flexible storage medium. When an object is stored in a collection it is implicitly cast to an Object. (Recall that Object is the base class from which all other classes are derived.) The methods used to store and retrieve Objects are collection-specific. For purposes of consistency, an object retrieved from a collection is returned as an Object. In order to perform a useful action on the Object, it must be recast to a reference of its original type (or its superclass).

The flexibility of Java collections can also be a disadvantage. For instance, since the collections are not type-specific, there is no type checking when an object is added. Consequently a collection of Strings can have a Thread object added to it without the compiler of Java runtime catching it.

The correct programming paradigm for code recovering objects from collections is always to catch ClassCastException.

The Stack Class

The Stack class implements a last-in-first-out stack of Objects. Objects can be "pushed" onto the top of the stack, then "popped" off. The first object to be pushed onto the stack will be the last one to be popped, and vice versa. To facilitate these operations, the methods push and pop are provided.

Popping an object off the stack will remove it from the stack. If you wish to examine an object on the top without removing it, then you can use the method peek().

In general, in order to examine the contents of the stack, you can use the search() method. This will look at the elements of the stack for the Object supplied as a parameter using the Object.equals() method as a means of locating it. If it finds it, the search returns its position in the stack.

If the return value is 1, this means that the Object is the topmost item in the stack. A value of 2 suggests the second place, and so on. A value of -1 suggests that the Object is not on the stack.

Both pop() and peek() will generate an EmptyStackExpression if there is no Object on the stack. You should always perform a boolean check with the empty() method when popping Objects from the stack.

Abstract Dictionaries

The Dictionary class is an abstract class that provides methods to store and retrieve an Object indexed by a key rather than an index. All of its methods are abstract. (It is effectively an interface, being "extended" rather than "implemented.") This means that the precise algorithms used to map and store keys are determined within the subclasses.

The simplest manipulations are put() and get(). The put() method places the Object into the Dictionary in an appropriate place based on the key that has been supplied. If another Object has already been stored using the same key, the new Object is stored with that key, and the put() method returns the Object previously stored there. The get() method returns the Object stored under the key supplied. In both cases, get() and put(), if the position associated with the key supplied is not occupied, a value of null is returned.

This means that neither the key nor the element can be null, since null is used to indicate that an Object was not found in that location. Attempts to use null will result in an exception.

Objects can be removed from the Dictionary with remove(), which returns the object associated with the given key and removes it. The current number of Objects in the Dictionary can be determined with size(); the isEmpty() method is a special case to test for a size of 0 (zero).

We know that abstract classes are not in themselves very useful, but the Dictionary class is the parent class of two other very useful classes: HashTable and Properties.

Properties Lists

The Properties class builds on HashTable to provide an interface to a set of key/value pairs, both of which are strings. The most obvious place this is used is for the list of system properties with System.getProperty(). Although the method names used to manipulate the property list are specific to the Properties class, they all use the HashTable methods to store and retrieve data.

The Properties class builds on HashTable to provide an interface to a set of key/value pairs, both of which are strings.

The no-argument constructor creates an empty property list. You can use put() to populate this list. The alternative constructor takes a default Properties list. If the requested key is not found in the current Properties object, then it is searched for in the default list. Note that the default list can itself have a default list, etc. It is not possible to change the default property list after construction.

The simplest thing to do with a Properties object is to query it with getProperty(). You supply the key, and it returns the matching value String (or null if not found). The alternative form allows you to supply a default value to be returned instead of null if the key is not found.

The Properties class also has a useful feature that allows you to save a properties list to an output stream and recover it from an input stream (typically a file). The save() method writes the list to the given stream. It is acceptable to supply a single-line comment that is placed at the start of the file. These properties can subsequently be read into a Properties object with load().

All of the properties can be recovered as an enumeration with the propertyNames() method or listed on a given print stream with list().

Random Numbers

The Random class permits the user to create and manipulate a pseudo-random-number generator. The argument constructor provides a random-number generator seeded with a value based on the current time. Alternatively, it is possible to set your own "long" seed value in the constructor or later with setSeed().

Once you have your Random object, you can generate uniformly distributed pseudo-random numbers of various types: int, long, float, double. In each case, the value generated will be between the minimum and maximum values for the given type, for example Integer.MIN_VALUE or Integer.MAX_VALUE.

If you do not wish to deal with all the overhead of object creation and destruction, it is possible to use the static method random() in the Math class. This returns a double value that can be converted into the desired type.

Example Using the Properties Collection Class

This program displays the system properties specified as input parameters or, if no input is given, all the system properties.

```
package java4cpp.collections;
import java.util.*;

public class SystemProperties {

        public static void main(String[] args) {
     if (args.length > 0) {
        for (int i = 0; i < args.length; i++) {
           System.out.println(args[i] + ": " +
              System.getProperty(args[i], "not found"));
        }

     } else {
        // dump all the system properties
        Properties sysProps = System.getProperties();
        Enumeration e = sysProps.propertyNames();
           while(e.hasMoreElements() ) {
              String propName = (String) e.nextElement();
                 System.out.println(propName + ": " +
                    sysProps.getProperty(propName));
           }
        }
     }
}
```

Compile the program: javac SystemProperties.java
Run the program: java SystemProperties

When there are no input arguments, the output describes the native environment on which the program has run and the implementation of the JVM used. The following output or one similar will be displayed if you run this program.

```
user.language: en
user.name: prithvi
java.home: /usr/local/jdk1.2.2/bin
file.encoding.pkg: sun.io
java.version: 1.2.2
file separator: /
line.separator:
user.region: US
file.encoding: 8859_1
user.timezone: EST
path.separator: :
```

Summary

We have examined some of the collections that are part of Java. Collections can be organized into abstract structures such as sets, bags, lists, stacks, trees, queues, hash tables, and many others. When selecting a collection class, we have seen that there are abstract classes that must be extended or that simply use one of the data structures already encapsulated by any of the core collection class (such as Properties in the example above). No matter which you choose, there is a greater chance that your code will be more elegant.

The overhead of object creation and destruction can result in a possible performance degradation, but if used wisely this too can be ameliorated.

the tclsh spot

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

The last Tclsh Spot article showed how to write a simple HTML robot that gets the current price of a stock using the HTTP package and some simple regular expressions.

The final code looked like this:

```
package require http

foreach symbol $argv {
    set url "http://www.newsalert.com/free/stocknews?Symbol=$symbol"

    set id [::http::geturl $url]
    set data [::http::data $id]
    regexp {arts\?Symbol=(.+?)">(.+?)<} $data match symbol pric"
    puts "$symbol last traded at $price"
    }
```

This is useful information, but there is a lot more info in the report from *newsalert.com* that it would be nice to get, and the regexp command seems like a good tool to use.

The interesting part of the page from *newsalert.com* looks like this:

```
<td align="left" class="symprice"> <a"href="/bin/charts?Symbol=SUNW">142 </a>
$
<td align="left" class="indexNumUp"> <img src="/gifs/uparrow2.gif" width="9"
                    height="8" alt="" border="0">6 </td>
<td align="left" class="symprice">4.4 </td>
<td align="left" class="symprice">12/3 </td>
<td align="left" class="symprice">138 15/16 </td>
<td align="left" class="symprice">143 3/8 </td>
<td align="left" class="symprice">138 7/8 </td>
<td align="left" class="symprice">11,781 </td>
```

The obvious pattern is that all the interesting data is in between a > and a < symbol after the charts?Symbol= string.

A simple brute-force approach to this regular expression would be to match each < *stuff* > Data < *stuff*> pattern with a regular expression something like this:

```
<[^>]+>([^<]+)<[^>]+>
```

Another new feature with the 8.1 and 8.2 Tcl interpreter is the -expanded flag. The -expanded flag causes the regular-expression parser to ignore whitespace and comments.

This regular expression matches a < symbol, any other characters except a greater-than symbol up to the first > symbol (the <td align left . . . > tag), then any characters except a less-than symbol (the data) until the next < and any characters except the greater-than until the next > (the </td> tag).

This style of regular expression is familiar to most folks who have used regular expressions with sed or fgrep, and it will work with all revisions of Tcl.

The regular-expression code was rewritten for Tcl 8.1. Among other improvements (such as support for Unicode), a question mark after a quantifier symbol (the + or *) will change the regular-expression parsing behavior from using the maximum number of characters to match a regular expression to using the minimum number of characters to match the expression.

This lets us simplify the previous pattern to this:

```
<.+?>(.+?)<.+?>
```

This regular expression matches a < symbol, any other characters up to the first > symbol, then any characters up to the next < and any characters until the next >.

We can concatenate as many copies of this pattern as we need to collect the change, percentage change, high, low, etc. This is conceptually simple but creates a long, incomprehensible regular expression.

Another new feature with the 8.1 and 8.2 Tcl interpreter is the -expanded flag. The -expanded flag causes the regular-expression parser to ignore whitespace and comments. Thus, instead of an ugly long, regular expression, we can write

```
regexp -expanded {arts\?Symbol=(.+?)"> # Get the symbol
    (.+?)                              # and the last sale price
    <.+?>                              # Close the Href
    .+?
    <.+?>                              # Close the Table definition
     .+?
    <.+?>                              # Start a table definition
    .+?
    <.+?>                              # The image declaration
    (.+?)                              # Absolute change
    <.+?>                              # Close table} $page m symb pric chg
```

This is readable, but still longer than seems necessary.

The Tcl regular-expression engine can be declared with repeating atoms. An atom can be a single character, or a regular expression enclosed in parentheses. The number of times the pattern can be matched is declared by following the atom with a value inside curly braces.

{val}	Match the pattern exactly val times.
{val,}	Match the pattern at least val times.
{val,max}	Match the pattern at least val times, and at most max times.

This lets us shorten the regular expression by removing the repeated pattern of a tag followed by unnecessary characters.

```
regexp -expanded  {arts\?Symbol=(.+?)">   # Get the symbol
                  (.+?)                   # and the last sale price
                  (<.+?>                  # Close the Href and Close Table column
                  .+?){3}                 # start next table column
                  <.+?>                   # The image declaration
                  (.+?)                   # Absolute change
                  <.+?>                   # Close table} $page m symb pric dummy chg
```

The dummy variable is there to catch the part of the string that's matched by the duplicated regular expression.

We won't be using that data, so there is really no need to collect it.

A regular expression can be made "non-capturing" by following the left parenthesis with a question mark and colon, instead of continuing with the regular expression.

```
regexp -expanded    {arts\?Symbol=(.+?)"π>          # Get the symbol
                    (.+?)                          # and the last sale price
                    (?:<.+?>                       # Close the Href and Close Table column
                    .+?){3}                        # start next table column
                    <.+?>                          # The image declaration
                    (.+?)                          # Absolute change
                    <.+?>                          # Close table} $page m symb pric chg
```

This technique makes a fairly comprehensible regular expression, but we've only gotten as far as the price change. The code doesn't handle high, low, date, volume, etc.

More fields can be added to the regular expression by just extending this pattern, but that will get long.

We can use the match count descriptor to extract successive fields from a regular expression into variables by using a loop like this:

```
array set varnames {1 chg 2 pct 3 time 4 open 5 high 6 low 7 volume}
for {set i 1} {$i < 8} {incr i} {
regexp -expanded "  arts.Symbol=(.+?)\">        # Get the symbol
                    (.+?)                      # and the last sale price
                    (?:<.+?>                   # Close the Href and Close Table column
                    .+?){3}                    # start next table column
                    (?:<.+?>                   # The image declaration
                    (.+?)                      # Absolute change
                    <.+?>.+?){$i}"             $page m symb pric $varnames($i)
}
```

This is fairly short and uses some nice new features of the regexp command. The only problem is that it parses the full HTML page to extract each and every item. That's only eight passes, but still . . .

The data that we want to get from this page are the only characters not inside a tag. If we just strip the tag info away from the text, we'll be left with the data we want.

There are other Tcl commands that use the regular-expression engine. One of these is regsub.

Syntax: *regsub ?options? expression string subSpec varName*

regsub	Copies *string* to the variable *varName*. If *expression* matches a portion of *string*, then that portion is replaced by *subSpec*.
options	Options to finetune the behavior of *regsub*. May be one of:
-all	Replace all occurrences of the regular expression with the replacement string. By default only the first occurrence is replaced.
-nocase	Ignores the case of letters when searching for match.
--	Marks the end of options. Arguments that follow this will be treated as regular expressions, even if they start with a dash.

expression	A regular expression that will be compared to the target string.
string	A target string to which the regular expression will be compared.
subSpec	A string that will replace the regular expression in the target string.
varName	A variable in which the modified target string will be placed.

In this case, we can use the regexp command to strip away most of the HTML page, leaving us with several lines of HTML data that describe a single row in the table. Our script can use the regsub command to strip the tag information out of that string, and finally convert the multiple lines of data into a list with the split command.

```
regexp -expanded "arts.Symbol=(.+?)\"> # Get the symbol
    (.+?)<tr>"  $page m symb data
regsub -all {<.+?>} $data {} lines
set dataList [split $lines \n]
```

This is a fairly small set of code for parsing the information out of these HTML pages. A robot using this parsing code would resemble this:

```
package require http

::http::config -proxyhost 56.0.0.2 -proxyport 8000

puts [format \
    {%-5s %-9s %-7s %-5s %-6s %-9s %-9s %-9s %-8s}\
    symb last change pct date open high low volume]

foreach symbol $argv {
    set url "http://www.newsalert.com/free/stocknews?Symbol=$symbol"

    set id [::http::geturl $url]
    set page [::http::data $id]

    regexp -expanded "arts.Symbol=(.+?)\"> # Get the symbol
       (.+?)<tr>"      $page m symb data
    regsub -all {<.+?>} $data {} lines
    set dataList [split [string trim $lines] \n]

    puts [eval format \
       {{%-5s %-9s %-7s %-5s %-6s %-9s %-9s %-9s %-8s}} \
       $symbol $dataList]
}
```

This robot will generate output resembling:

```
symb   last       change    pct    time  open        high      low        volume
sunw   142           6      4.4    12/3  138 15/16   143 3/8   138 7/8      24
hp     23 7/16     -1 1/16  -4.3   12/3   24 1/2      24 1/2    23 5/16   176
```

Again, this robot can be put into a crontab entry to put stock quotes into your mailbox.

But now that I can collect daily quotes, I want to analyze and view the data. The next article will discuss saving the data and using the BLT graph widget to view it.

effective perl programming

CGI Barbie says, "Programming is hard!"

by Joseph N. Hall

Joseph N. Hall is the author of *Effective Perl Programming* (Addison-Wesley, 1998). He teaches Perl classes, consults, and plays a lot of golf in his spare time.

<joseph@5sigma.com>

CGI programming in Perl has brought a lot of new programmers into the Perl and UNIX world. While I'm always happy to see more people learning the fine art of programming, CGI isn't the gentlest introduction to the art. Nor is it always the safest. It is notoriously difficult to write programs that safely take user input and use it to perform file operations and/or run commands.

In this column, I examine an application that is similar to one I have had my students write in my CGI-programming classes. What it does is simple, yet it turns out to be surprisingly difficult to make it robust and secure from misuse. But isn't life always this way?

A "Simple" Directory Lister

Our program is a relatively simple one that lists the contents of directories and displays the contents of text files in those directories. I'll give it to you in pieces with a bit of explanation in between.

```
#!/usr/local/bin/perl -w
use strict;
use CGI qw(:standard);
use URI::Escape;              # for uri_escape
use HTML::Entities;           # for encode_entities
use Cwd;                      # for cwd
```

We'll run the program with warnings (-w) turned on. The CGI, URI::Escape, and HTML::Entities modules are CGI-related. The Cwd module gives us a portable way of converting the current directory to an absolute pathname.

```
my $root_dir = ".";
die "couldn't read the root dir" unless -r $root_dir and -x $root_dir;
chdir $root_dir or die "couldn't change to the root dir: $!";
my $script_name = script_name;
my $orig_cwd_dir = cwd;
```

The program lists the contents of $root_dir by default. The "specs" for this program say that we should be able to list the contents of directories below but not above $root_dir. script_name() is a function from CGI.pm that returns the URL of this script. cwd() is a function from Cwd that returns an absolute path for the current working directory, like the pwd command.

```
sub errorpage {
    my $msg = shift;
    print header,
      start_html, h1('Error'), p(encode_entities($msg)),
      p(a({-href => $script_name}, "go back to $script_name")),
      end_html;
    exit;
}
```

The program defines a subroutine called errorpage used in the event of fatal errors. It generates HTML around an error message. Here, as well as in other places below, we use the encode_entities function from HTML::Entities to escape HTML entities that might appear in the output, so that if an error message contains an HTML metacharacter like "&" it will be converted into its escaped equivalent ("&amp;" in this case) before it is output. A link to $script_name gives users an alternative to the back button.

I usually prefer to accumulate and save output while there is still interesting work left to perform. If the program emits half a page of HTML and then encounters a fatal error, the resulting unfinished HTML will look ugly.

```
my $dir = param('dir');
$dir = '.' if !defined($dir) or $dir eq '';
my $file = param('file');
```

The program takes two CGI parameters. If a file parameter is present, the program will treat it as the name of a text file and display the contents of that file. The dir parameter, if present, specifies the directory for the file parameter. If no file parameter is present, the program lists the contents of the dir directory, or $root_dir if there is no dir either. The dir parameter is relative to $root_dir.

```
if (defined($file) and $file ne '') {
    chdir $dir or errorpage "can't change to directory $dir: $!";
    open F, $file or errorpage "couldn't open $file: $!";
    print header('text/plain');
    print "file: $file\n\n";
    print <F>;
```

Here is the code that checks to see whether there is a file parameter. If there is, we use it as the name of a file in the current directory (set from the dir parameter) and display its contents as a text/plain document.

```
}   else {
    my $dir_uri = uri_escape $dir, '\W';
    chdir $dir or errorpage "can't change to directory $dir: $!";
    my $cwd_dir = cwd;
    opendir DIR, '.' or errorpage "can't open directory $cwd_dir: $!";
    my @files = sort readdir DIR;
    closedir DIR;
```

If there's no file to display, we display the contents of the dir directory (or .) instead. The uri_escape() function changes URI metacharacters like "+" and "%" into URI-escaped equivalents like "%2B" and "%25" – we'll need this when we construct HTML anchors in our document, later. $cwd_dir is an absolute pathname for the directory in the dir parameter.

```
my @table;
push @table, "<table>\n";
push @table, "<tr><td><b>Filename</b></td>
    <td><b>Size</b></td><td><b>Last Modified</b></td></tr>\n";
```

Output gets "spooled" into an array called @table. I usually prefer to accumulate and save output while there is still interesting work left to perform. If the program emits half a page of HTML and then encounters a fatal error, the resulting unfinished HTML will look ugly, so I don't emit HTML until the very end.

```
if ($orig_cwd_dir ne $cwd_dir) {
    my $up_dir = $dir;
    $up_dir =~ s#/[^/]*$##; # lop off last /dir
    push @table, "<tr><td><tt><a href=\"$script_name?dir=" .
        uri_escape($up_dir) . '">Up one directory</a></tt></td>' .
        "<td></td><td></td></tr>\n";
}
```

This code adds a link to the directory "above" if we are browsing a directory other than $root_dir.

```
foreach my $fn (@files) {
    next if $fn eq "." or $fn eq "..";
    my $fn_enc = encode_entities $fn_enc;
    my $fn_uri = uri_escape $fn, '\W';
```

```
        my ($a, $aa) = (-T $fn) ?
            (qq(<a href="$script_name?file=$fn_uri&dir=$dir_uri">), "</a>") :
            ('', '');
        my ($b, $bb) = (-d $fn ) ?
            (qq(<b><a href="$script_name?dir=$dir_uri/$fn_uri">), "</a></b>") :
            ('', '');
        my $bytes_k = int(((-s $fn) + 1023) / 1024);
        my $mod_time = localtime((stat $fn)[9]);
        push @table, "<tr><td><tt>$a$b$fn_enc$bb$aa</tt></td>
        <td><tt>${bytes_k}k</tt></td><td><tt>$mod_time</tt></td </tr>\n";
    }
    push @table, "</table>\n";
```

This block of code traverses the list of files that we obtained with the readdir earlier and creates a line of output for each of them. We display the filename (HTML-escaping it so that filenames like &amp; will display correctly) and its size and modification time. We also create links that can be used to view text files and subdirectories. Filenames in the link URLs have to be URI-escaped so that links to files with names like a+b work correctly.

```
    print header,
        start_html($cwd_dir),
        h1("Directory of ", tt($cwd_dir)), "\n",
        @table,
        end_html;
    }
```

Finally, we emit our HTML and are done. Or are we?

Closing Up the Holes – They're Everywhere!

Despite its relatively simple construction and user interface, this program is filled with not-so-obvious flaws. The worst among them is that it can be used to execute arbitrary UNIX commands on the server via Perl's open operator. Notice that when we opened the file above we just said open F, $filename. What if we run this program with the query string "file=cat+/etc/passwd|"? Well, the open becomes open F, "cat /etc/passwd|", which opens a pipe to the cat command and displays the contents of /etc/passwd! I'm not fond of the syntax of the open operator and think that it should never be used to process a user-specified filename in a CGI program. I prefer to use Fcntl and open the file with sysopen instead:

```
    sysopen F, $file, O_RDONLY or errorpage "couldn't open $file: $!";
```

We now no longer have to worry about metacharacters like "|", because sysopen only knows how to open files.

The program also has problems in that it can display files outside $root_dir and its subdirectories. It takes a bit of work to fix it so that it doesn't violate the spec that requires it to operate in just that part of the directory tree. First, before we do the sysopen, let's perform some other checks to disallow things like file=../../../../../../etc/passwd:

```
    if (index($file, '/') > -1) {
        errorpage "filenames containing / are not allowed";
    }
    errorpage "no such file $file" unless -f $file;
    errorpage "$file doesn't look like a text file" unless -T $file;
```

That takes care of the filename, but what about the dir parameter? It has the same relative path problems, so we have to work on that too. The following code would go after my $dir = param('dir'):

This program is filled with not-so-obvious flaws. The worst among them is that it can be used to execute arbitrary UNIX commands on the server via Perl's open operator.

A poorly written CGI program can be a source of exploits for local users.

```
if (defined($dir) and $dir ne '') {
      errorpage "directory starting with / not allowed" if $dir =~ m#^/#;
      errorpage "directory can't have .." if index($dir, '..') > -1;
      errorpage "no such directory $dir" unless -d $dir;
}
```

This eliminates the obvious cases. However, if the directory we are browsing has a symlink to some other directory, we'll be able to use the symlink to look at a directory outside the tree. To fix this, after if ($orig_cwd_dir ne $cwd_dir), put:

```
errorpage "directory $cwd_dir doesn't seem right" unless
      index($cwd_dir, $orig_cwd_dir) == 0;
```

Because $cwd_dir is supposed to be "below" $orig_cwd_dir, $orig_cwd_dir should be a prefix of $cwd_dir. This will abort the execution of the script if it's not. We also want to avoid generating anchors to bad places in the first place:

```
my ($b, $bb) = (-d $fn and not -l $fn and -x $fn and -r $fn) ?
      (qq(<b><a href="$script_name?dir=$dir_uri/$fn_uri">), "</a></b>") :
      ('', '');
```

This tests for symlinks (-l $fn), as well as ensures that directories are executable and readable.

Additional Paranoia

After spending hours or days on a program intended to run in the slightly weird environment of CGI, it's easy to forget that the same program can be run from the command line. A poorly written CGI program can be a source of exploits for local users. Hopefully you don't have any bad eggs logged onto your Web server, but who knows? One thing I like to do in my more complex CGI programs is to test to see whether the program is being run by the expected user. This is very important for setuid programs and not a bad idea for other programs. I usually do this with a BEGIN block before everything else in the program (even the use directives), so that the check will occur before any other code in the program runs.

```
BEGIN {
      if ($< != getpwnam('joseph') and $< != getpwnam('nobody')) {
          die "This program may not be run by user $<";
      }
}
```

Even though this program doesn't run any subshells or commands (not anymore, anyway), it's not a bad idea to turn on Perl's tainting feature. When tainting is turned on, Perl will yield a fatal error whenever user-supplied input (from files, standard input, environment variables, or a variety of other sources) is used as part of a subshell, command, or file operation. Tainting also invokes a number of other checks, such as checking the execution path for insecure directories. Tainting is automatically turned on for setuid programs, but you can activate it explicitly with the -T command-line switch:

```
#!/usr/local/bin/perl -Tw
```

With tainting turned on, we have to clean up the execution path:

```
$ENV{PATH} = "/bin:/usr/bin";
$ENV{BASH_ENV} = "/bin:/usr/bin";
```

The prohibition on file operations using user-supplied data includes the chdir operator. To use the dir parameter with chdir, we have to untaint its value. To untaint data, you perform a pattern match and extract portions with subexpressions (parentheses). Data captured in the subexpressions is untainted:

```
($dir) = $dir =~ /([^\0]*)/;
```

The purpose of untainting is to eliminate metacharacters that might be the source of some unwanted or unintended behavior – often, shell metacharacters in commands, but others as well. All manner of characters are valid in directory names, but I've used this as an opportunity to ensure that the directory name does not contain nulls.

Tainting doesn't actually help this program very much, but it's a good feature to use so that it'll be harder to make silly mistakes if you pass the code along to someone else or modify it yourself in the future.

Programming Really Is Hard

My point in presenting this example hasn't been to teach the gentle art of CGI programming. Rather, it has been to illustrate that although the concepts behind CGI programming are relatively simple, it can be difficult to eliminate unwanted side effects in programs that deal with user input. Your system's CGI users may be some of its least-experienced programmers, yet the issues that they have to confront when dealing with access to files and subshells can be tricky even for very experienced developers. You should never underestimate the ability of seemingly innocuous user-written CGI programs to create security risks for your servers.

For more about security issues in CGI programming, see the World Wide Web security FAQ at *<http://www.w3.org/Security/Faq/www-security-faq.html>*. See the Perl perlsec man page for more about tainting, and the perlfunc man page for more about Perl's strangely (and sometimes dangerously) versatile open operator. A commented version of the entire program referred to in this article is available in *<http://www.perlfaq.com/examples>*.

Although the concepts behind CGI programming are relatively simple, it can be difficult to eliminate unwanted side effects in programs that deal with user input.

the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CEO of Network Flight Recorder, Inc. He likes cats: they are complex yet manageable. When he's not working 10-hour days he plays console games and pursues too many hobbies for his own good.

<mjr@nfr.net>

As I write this, security is front-page news because Yahoo.com, Amazon.com, CNN.com, eBay.com, and others are under concerted denial-of-service attack from terrorists whose motivations and identities are, at present, unknown. As someone who runs a business, I can tell you how terrifying it is to contemplate being shut down by someone you can't identify or block. This kind of thing is about as elegant as a drive-by shooting and shows that, as a society, people couldn't resist bringing the worst aspects of "the real world" into cyberspace. I wonder if the perpetrators think it's funny. What motivates them? For that matter, what motivates the people who developed and released the tools the attackers are using? And what drives the virus writers? Come on, guys, it's not cool – you're hurting people.

Once again, security analysts and network managers are being presented with an extremely challenging problem that we really don't know how to solve conclusively. It's going to cost the Internet and its users millions of dollars in wasted time, lost revenue, and ulcers over the next few years. I've been happy, generally, with the media's response to this event, as compared to past problems. This time, at least, they are not saying it's being done by "brilliant" misunderstood "whiz kids" or anything like that. Perhaps the hackers made a big mistake going after CNN.com – media's "nudge-nudge-wink-wink" attitude toward hacking has done a lot to make it seem sexy and cutting edge. Biting the hand of the media is an error of judgment sure to get you slapped, as many politicians have learned. Perhaps their irritating the media will help adjust the press's coverage. Fundamentally, these are social problems, not technological problems, and are best solved by social means. The media has to take its part, since it is, arguably, the "voice" of society. This kind of nonsense isn't the work of "brilliant whiz kids"; it's the work of socially maladjusted losers.

Boundless Overruns

Lately, the problem of buffer overruns in code has become serious. It seems that virtually every application designed to work on a network has suffered some kind of buffer overrun recently. If an application hasn't, it's probably just because the hackers haven't looked at it – yet. Obviously, it's not a *new* problem, really – like many security problems, it's one we've known about for a very long time. I guess circumstances have pretty much conspired to let us ignore it for many years, but now the grace period is emphatically over. In the early 1990s I wrote a couple of firewall products; recently, in order to torture myself, I went back, pulled the source out of my archives, and did code reviews of some modules. When I wrote them, I know I was trying to be pretty careful, but in several places I was not careful enough. At least I tried, I suppose. What about the developers who don't know they need to be careful? What about the developers of applications that aren't believed to be security-critical? My guess is that the vast majority of networked applications are rife with security bugs.

What can we do about it, if anything? Recent flaws in "mission-critical" software seem to point out just how little progress we have made in securing our critical code, yet every day we field new applications, most of which were developed with only a cursory thought to security. I suspect that, as the software market continues to heat up, the environment will get worse, faster. We're seeing start-ups going from zero to product ship in months, and from product ship to installed bases of hundreds of thousands of users in weeks. About a year ago, I had a scary realization:

Today's toy is tomorrow's critical business tool.

We can't predict which ones will be "big," but a significant percentage of the buggy crud that is being written today will become part of our "mission-critical infrastructure." For example, lots of firewall administrators block instant-messenger–type programs because of various security concerns. I remember trying that when HTTP first came out – who'd need that stuff? My prediction is that within a few years, one of those online messenger programs will be an essential business tool, and people will be using it to buy and sell stocks or broker mergers and acquisitions. Because of the huge legacy installed base that will be in place by then, it'll be too late to add security into the application's protocols. I now believe it is impossible to accomplish security in such an environment – all we can do is struggle valiantly on the sharp end of the hook.

The way we treat hackers and think about hacking is going to have to change.

In order to make the situation improve we'll need to somehow completely revamp the way in which we develop software, publish it, and distribute it. We'll also need social change – the way we treat hackers and think about hacking is going to have to change. Another step in the right direction would be to scrap most of our applications base and replace our core software infrastructure with something simpler and better designed. None of that is going to happen, possibly ever. I don't want to contemplate the kind of disaster we'd need in order to trigger the degree of social change that is necessary – it'll take some kind of massive "software Chernobyl" and a couple of iterations of overreaction before we ever progress in the right direction.

That's a kind of depressing view of the situation, I realize. I guess that the news about the massive denial-of-service attacks has put me in a pretty negative frame of mind about the state of security. This kind of nonsense cannot continue.

Some Suggestions

Let me try to be a bit more positive with some suggestions for simple things you can do to help improve the state of computer security from the comfort of your workstation. One is simple and has to do with bug fixes; the other is more of a management issue and has to do with documentation and design.

One area I don't think we've adequately tapped is intrusion detection in applications. That's because the only people who can meaningfully add application intrusion detection are the developers themselves. It's such an obvious idea that I'm really kicking myself for not thinking of it until recently; this is something we should have been promoting for a long time. If a security flaw is identified in your code, don't just put a patch in place to fix the problem; put a patch in place that identifies attempts to trigger the problem. Log the attempt and block it. If we all did this in our code, we'd begin to collect extremely detailed information about who is trying to exploit specific vulnerabilities. This approach would also be pretty safe against false positives generated by legitimate scanning tools – such tools typically don't try to exploit the actual vulnerability, but simply test for its existence. It's free, cheap, utterly reliable intrusion detection with no performance cost. About the only drawback I see to this idea is that some bodies of code would be dramatically enlarged. Not coincidentally, a lot of them are already hugely bloated – so what'd be the harm of adding a few thousand more lines of code to a Web server, browser, or mail-transfer agent? There'd be a bit of a maintenance cost, but it's not too large.

Second, please stop making your applications identify themselves obviously over a network. While it's essential that protocol negotiations still work, it is not necessary to announce what version of a particular server you are running. Right now, huge numbers of applications effectively paint a target on their backs by announcing to the world at large, "Hello! I am whateverdV1.22!" Even the lowliest script kiddie can use such

An interesting thing happens when you hit a security-illiterate person with a tool that makes them realize how often they are probed and examined by hackers: they get furious.

obvious targeting information to search rootshell.com for vulnerabilities. Without identifying information, they'll have to use more sophisticated techniques such as stack-fingerprinting approaches applied to applications. That'll be okay because when someone starts fingerprinting your applications, that will be easier to detect (especially from within the application) and is more clearly hostile action. Remember, the more intrusive we can force probes to become, the easier they are to detect and the harder they are to laugh off as "innocent curiosity."

Imagine what would happen if a majority of applications were self-instrumented security alarms. Within an hour of installing your new Web server, you would begin receiving notices from the server of all the attempts being launched against it. My guess is that larger sites would get dozens of warnings a minute. The great part is that the hackers wouldn't really be able to tell if you were running a version of a server that was tattling on them, or a vulnerable version. Let's just make it a little harder for them, shall we?

Of course, I have a stealthy agenda in proposing this concept: it would stun people if they realized how often they come under attack. An interesting thing happens when you hit a security-illiterate person with a tool that makes them realize how often they are probed and examined by hackers: they get furious. If you're careful about how you do it, they don't get mad at you, they get mad at the hacker, complain to their ISPs, the police, the press, etc. That's the level of attention and frustration that leads to "there ought to be a law" thinking – a necessary part of the process of adding pain to cause a backlash.

Another thing developers can do is document whether their applications are safe for use in a privileged context. This is a minor point, but it might help. I've been amazed to see developers employ tools that had huge security flaws as components of critical applications. "But it comes with the system," they cry when someone points out the problem. That complaint is usually followed by "It's too late now, we have to push this into production and it already works." No, grasshopper, it merely appears to work. I'd like to see applications like ftp come with a warning label on them saying, in effect, "Not for use over public networks." We have to raise awareness of these constraints before the point at which system developers decide what components they will use to build their architectures. I am reminded of BSD's approach to getting rid of the obsolete and evil `gets()` function call: they modified it to print a message to the screen when the function was called: "WARNING: This program uses `gets()` which is insecure." Perhaps we need to have applications like ftp check and see if the destination is off the local subnet and print a warning like: "WARNING: This program uses insecure networking and should not be used off your local subnet." On the surface that seems like a decent idea, except that, of course, virtually every application would be printing terrifying warnings. Such warnings are more food for the eventual backlash.

One thing that's always frustrated me is the relative ease with which these broken applications could be upgraded. The only thing stopping us is the Evil Demon of Backward Compatibility, who grows stronger every day as tens of thousands of new users flock to the Internet. Eventually, I believe we will need to slay the demon by scrapping whole chunks of redundant protocols and layering them atop newer, better-designed ones. For example SSL, with whatever warts it may possess, is still a much better protocol, security-wise and in terms of port usage, than ftp. Someone could develop a new version of "ftp" that had the same user interface but that used SSL as an underlying protocol. Many users would never notice. The same could be done with telnet, rlogin, rsh, etc., etc. This would improve security not just in the obvious way (getting rid of passwords crossing the Net in the clear) but by reducing the amount of redundant networking code that can be implemented incorrectly. Perhaps this might be something the open source movement could take on as part of general legacy code cleanup . . .

The Weather Sure Is Big Out

Well, I fear I may have rambled a bit. In my last column I promised that I'd try to keep things more technical in this column. I hope I haven't completely reneged on that promise. From where I sit, I'm becoming convinced that security isn't really a technical problem – or, more precisely, that the technical component of security is vanishingly small compared to the political, management, and financial components of the security landscape. Large weather out; looks like it's going to rain . . .

In the same vein as last column, I will propose another contest (prize will be a nifty keen "Network Police" jacket with a T-shirt for the runner-up). The topic of this contest is "Where do packets go when they die?" Be brief.

an interview with mudge

by Mudge

Mudge is VP of R&D for @Stake Inc.

<mudge@L0pht.com>

[Editor's Note: *Mudge was formerly CEO and Chief Scientist for the L0pht, which recently merged with @Stake. Rob Kolstad interviewed Mudge via email during the week of February 7, 2000.*]

Rob: I know that many people don't have a clear picture of L0pht Heavy Industries. Can you tell us about the organization's (former) function?

Mudge: The L0pht, now @Stake's R&D leg, started out as a loose organization of people in the same geographic area. The common bond was that we were (and still are) driven by our interest in technology and security. We chipped in to get a storage space for the various computers, peripherals, and components that we had all amassed over the years. That was back in 1992 and was the first physical manifestation of the L0pht.

We promptly started setting up controlled test environments and creating a laboratory for development and testing. The one thing that was really lacking was an established body of work on network security. We decided that one of our goals would be to create such information and disseminate it. This turned into the collections such as the Whacked Mac Archives, the Black Crawling Systems, and the Advisories we released. We tried to document our research methodologies and results when looking at new technologies or security areas.

What we ended up becoming was the unofficial consumer watch group on network and computer security. We were not owned by a particular political group or vendor so we could speak our mind – and people could take what we said at face value. There was no hidden agenda of "Are they trying to sell product, promote a service they are offering, or sway people's alliances towards a particular angle?" All we wanted to do was to make sure that mistakes we discovered were being made were fixed and that others could

We are always trying to look at problems that, when solved, will offer improvements and longer-term solutions, not just stopgap fixes. These forays into emerging technologies farm out nuggets, gems, and offerings to the general professional-services organization.

learn from them rather than continuously repeat them. We feel that we succeeded there.

Rob: How did you develop such a strong background in security?

Mudge: All of us are perpetually curious about the world around us. As such we strive to understand how things work, fall apart, and can be improved on. This is something that must have been instilled early on in life as all of us seem to have it at the L0pht – er, @Stake R&D labs – no matter what it is we are working on. Art, music, cars, engineering, programming, etc. Much the way some people are avid crossword-puzzle enthusiasts, we see everything that is presented to us in as many possible angles and lights as we can.

Rob: Going forward, L0pht has become the R&D leg of @Stake. Can you describe the timing and new business functions? How will you enable the company to make money?

Mudge: The L0pht has been a real corporation for the past several years. A lot of people did not realize this. We started to see a trend in the larger organizations recognizing the value and talent that we had amassed. Many of the larger companies attempted to either purchase us outright or exclusively license all of our technologies. We declined over and over again. Finally, we decided to go out and see if there were any other companies that had the same long-term goals and vision as ours. That is when we found @Stake. We were looking for people who understood the value of being largely above reproach – i.e., no hidden agendas when dealing with customers or clients. We were looking for people focused not only on the tactical cleanup of problems once someone else points them out but also in providing strategic solutions to involve security at the beginning. We believe we found them.

The R&D group drives value in the custom research and the few extreme jobs that we take on for clients. We are always trying to look at problems that, when solved, will offer improvements and longer-term solutions, not just stopgap fixes. These forays into emerging technologies farm out nuggets, gems, and offerings to the general professional-services organization. We have known how to reap the benefits from our R&D work but were previously unable to figure out how to scale. That's where the @Stake management team comes in.

John Rando, in particular, is amazing. This is, after all, the man who was in charge of Digital Equipment Corp.'s services organization worldwide ($8 billion revenue, 25,000 employees) and then subsequently all of Compaq's. He is generally recognized as the examplar of how to run a services organization. Of course, just to make sure we had an overabundance of techie know-how, we snagged Dan Geer as our CTO.

Rob: How much does a good security audit cost, anyway?

Mudge: My belief is that most people do not know what a good security audit is in the first place. Many believe that a security audit is comprised of scanning your systems to look for known holes that have been posted to bugtraq or other security mailing lists (and often by places such as the L0pht). This is a reactive mechanism for dealing with security. Can we do this? Yes. Can we do better? Yes.

The first thing a good security audit should start out with is a solid understanding of what the customer's current business is and what they need to accomplish in the future. Only then can one weigh the risks and benefits of particular practices against security. More important, the methods and mechanisms put in place to patch existing problems need to be in line with future directions. What good is it to put a fix in place that has to be yanked out later because it impedes your organization's capability to do business?

If you look at the distributed denial-of-service attacks that just took place, which would make more sense? Wait until we are where we are right now and then install committed access rates at the Web service provider and start thinking about filtering RFC1918 and IANA reserved networks, or to think about what the business is in the first place: offering extremely large-throughput Web services, thus coming to the conclusion that CAR along with anti-spoofing rules would make sense as due diligence.

Rob: Security is such a wonderful thing to sell because it's hard to measure. Do you have any means of measuring it? Or is that counterproductive, in the sense that one can never be secure enough?

Mudge: Ahh, that's the old way of looking at things. We are going to see a wonderful paradigm shift in how people look at security. It used to be a cost center. It used to be a situation where you had to justify the security budget and only the IT organization held the purse strings. Now security is a revenue creator. After all, if I can figure out a way of securely offering access to more internal components, I can drive business. It is the different business units that are working toward new models of conducting business, offering services, and therefore incorporating security. But this will only work when we start looking at security from a strategic vantage point and not a tactical one. The world is becoming more decentralized and distributed and as such the old security model of bandage after the fact becomes unscalable.

Rob: Thanks!

the magic, art, and science of customer support

by Christopher M. Russo

Chris Russo manages engineering at GTE Internetworking. His focus continues to be satisfying the customer – whether that be a Web developer, a system administrator, or an end user.

<crusso@3rdmoon.com>

Part II: Priorities of Customer Support

The art of customer support is a critical topic that remains largely unaddressed in the world of systems administration. Even though it's often disregarded, misunderstood, or simply not thought of, good customer support is absolutely critical in any support organization, be it a back-end server room or a desktop-support group. This article is the second in a series exploring the nuances of the customer-support focus and helping sysadmins to improve their organizations and themselves.

Why does a customer choose one service over another? It seems like a relatively simple question on the surface, doesn't it? After all, customers will opt for your service because you have what they need and are better than the other guy, right? Or perhaps you're simply the only shop in town, as in the case of the very successful local pet store across town from me.

If you think about it for a while, you will realize that there are certain core components in the decision to select one service over another. These components can vary somewhat with your industry, your customer base, and other modifiers. For the most part, however, the base components all revolve around the same basic ideas: selection, courtesy, response time, turnaround time, convenience, and cost.

Let's quickly examine each of those components:

- *Selection* is simply the availability in your organization of the service or product that the customer is seeking. Sometimes intangibles such as know-how, wisdom, and intelligence fall under selection. For example, a customer coming to a computer store may want an adequate selection of parts, but sometimes he is also looking for a good selection of information to assist in the purchase. This can be a little sticky, so watch it carefully.
- *Courtesy* is fairly self-explanatory. It is simply the quality of being kind, considerate, and friendly to the people who have come to you or your organization for a service or product.
- *Response Time* is the speed with which you acknowledge that the customer requires a service or product. It is not the fulfillment of the need for the service or product, but, rather, the way in which you let customers know that you are aware of them and their needs, regardless of other issues. It is the time until the customer knows that he'll be attended to as soon as your "shop" has the time and resources to do so.
- *Turnaround Time* is the speed with which you actually fulfill the needs of your customer. A fast turnaround time means that the customer gets what she was looking for quickly and can move on to other things in short order.
- *Convenience* is simply the ease with which customers can obtain your products and services. It is a summary of every element of your product or service that makes it simple for your customers to choose you over another similar vendor. Propinquity to your customers, the availability of multiple shops with ample parking, a good store

layout, and a Web presence are all examples of points that make your shop more convenient.

- *Cost* is just that. It's how expensive or inexpensive your products or services are. If you charge $1,000 for a Ping-Pong ball, you are expensive (and barking mad). If you charge $.01, you are inexpensive (and probably seeking light counseling).

Now that you know what we are referring to by "core components of customer support," let's go back and talk a little more about my local pet store – we'll call it The Pet Stop.

The Pet Stop is a nice little shop. Its size is probably 1,200 square feet or so. It has four aisles and carries a decent selection of pretty much anything you could need for your pet. While it doesn't have every possible brand or variant of every possible item, the owner is very finicky and tends to select the best items from the available range, in order to optimize his space. Because of this, you are usually safe in choosing pretty much anything on his shelves.

He also carries a few small mammals and birds, an excellent selection of freshwater fish, and a decent selection of marine fish as well. All of the animal cages and fish tanks are very clean and neat, as is the entire store. With the exception of a few of the younger staff, the people who work at The Pet Stop are typically very knowledgeable. If asked a tough question, the more junior staff will readily admit that they do not know and will ask someone more senior for the answer rather than make a poor guess.

The people who work there typically are pleasant and will go out of their way to give you as much help as you need. There are always enough people in the store to help you within a reasonable period of time, and if they are not able to help you, they will at least stop, say "Hi," and explain that they are very sorry but they are very busy and will be sure to get to you as soon as they possibly can. The store is open from 9:00 a.m. to 9:00 p.m. Monday through Friday and until 6:00 p.m. on Saturday and Sunday. It's been the only decent pet store within a 15-or-so-mile range for about 20 years, and needless to say is very well established and well known in the area.

Sounds like a great little store, right? Well, no place is perfect. The store does carry some small animals and cats, but it does not carry dogs or turtles. Because of the size and high-rent location of The Pet Stop, the prices are also a touch high. The selection is a bit on the limited side, which causes problems if you are looking for a solution that is a little out of the ordinary.

Most unfortunately, the store owner has a tendency to snap at you if you are being, in his assessment, "stupid." "Stupid" behavior usually includes things like sticking your fingers in the macaw (big red parrot) cage, which is positioned strategically under a big red-and-white sign clearly indicating that this sort of behavior is a very bad idea. (If that bird can snap a whole walnut in half with his beak, what kind of chance do you think your fingers have?)

So now that we have some information that we need about The Pet Stop, why don't we try to determine why the store is such a success? Again, sounds pretty simple, right? It's a nice little local shop that carries nice things and has some nice people, right? Well, yes. There is, however, a significant amount of oversimplification in that sentence that would make it very hard for the owner to look at his organization and see what he could do to improve it. Let's help our pet-store owner out and try to break it down into a little more detail.

Let's start by taking each of the core components we mentioned earlier and see where our little pet store rates on them.

The store has some key items, but not many. It is a small place that does not specialize in any particular thing, though it does have a great selection of freshwater fish. Aside from the fish, the store is simply too small to rate anything other than an Average in Selection. It does, however, have some very knowledgeable folks and less-knowledgeable, but eager and helpful, junior staff who would certainly do their best to get you an answer. This gives The Pet Stop an Above Average rating on Selection of information and know-how (intangibles).

Clearly, the owner of The Pet Stop is a bit crotchety at times and would be happy to tell you to your face that you are being a moron, but as long as you aren't dunking your siblings in the piranha tank, he is agreeable and even pleasant. The other people in the store tend to be energetic and very friendly. Since the owner is usually present, and often feared, the store would overall score a Courtesy rating of Average.

The people working at The Pet Stop are always very good about acknowledging your presence and letting you know that they will be able to help you as soon as they possibly can. They rarely give an actual estimate of time, but they are always very attentive and keep you updated. The Pet Stop would probably score High on the Response Time scale.

The Pet Stop is usually adequately staffed, so it is very rare to have to wait longer than a few minutes for someone to help you get your fish or answer your question. While the employees are very good with managing the customers on the floor, The Pet Stop does not staff a full-time register person, so there are some, reasonably rare, occasions when a customer has to wait several minutes in the checkout line. Because of this slight shortcoming, The Pet Stop would probably only rate Above Average on Turnaround Time.

The store is clearly a big winner in the Convenience category, since for most of the surrounding towns the next option in pet stores is well over 15 miles away. The aisles are wide, and the items on the shelves are categorized and easily located. You can call them up and have them special-order items they do not usually stock so that they will be available on your next trip. Unfortunately, the parking lot is small and cramped, and the mall is difficult to get in and out of. The Pet Stop probably would score Above Average on the Convenience rating.

The Pet Stop is more expensive than most other stores for various reasons, including the high rent and small size of the store, which prevents it from lowering prices by increasing the volume of sales. Without question, "pet super-stores" offer prices much lower than an independent shop such as The Pet Stop. The Pet Stop probably would score a Below Average in the Cost arena.

Now that we have an idea of where The Pet Stop stands in each of our core components of customer support, let's put the results into a table so we can analyze our findings.

Core Component	*Pet Stop Rating*
Selection	Average
Selection (intangible)	Above Average
Courtesy	Average
Response Time	High
Turnaround Time	Above Average
Convenience	Above Average
Cost	Below Average

What, then, are the priorities of the customers of The Pet Stop? Well, it is somewhat hard to say from this table exactly what the order of importance is, but we can make some assumptions that cost, selection of tangible goods, and courtesy are not absolutely critical for the people who go there, because The Pet Stop really does not fare all that well in those categories. This seems somewhat odd, doesn't it?

On the basis of the information we collected, it would appear that we could effectively jack up prices and throw dead fish at our customers when they are being annoying (I'm sure you think I'm kidding), but so long as we're quick, know what we are talking about, and are the best game in town, we will still have great business! Yikes!

Well, actually . . . yes, that's somewhat true! This leads us to the obvious question of "Well, OK, so what are the most and least important priorities of customer support?" After all, you've spent your whole life waiting for an opportunity to set up a shop making serious money, all the while barking at your customers and hurling mini-flounders at their heads and keeping score on a big lit-up board with your employees . . . right?

Definition and clarification of the actual priorities is actually somewhat dangerous. There are most definitely priorities, and customers will unquestionably place more weight on one component than another, but it is very important to keep two things in mind.

The first is that all of the core components of customer support are extremely important, and the severe lack of any one of them can cause major problems for your organization. If you have no product, why will anyone come? If you take a herring and smack every single person who walks in the door across the face, why would anyone return? If you don't bother to acknowledge people's presence for hours on end, why will they wait? If you never help them at all, why would they bother to visit your shop? If you are a clueless dullard, why would they even talk to you?

All of the core components of customer support are extremely important, and the severe lack of any one of them can cause major problems for your organization.

The second thing to understand is that the difference in priority between any two of the core components is the equivalent in weight of perhaps several grains of sand. They are all, in fact, very close in priority to one another.

Assuming that we are not completely missing any major core components of customer support and that we have the ability and time to improve only a certain amount at a time, we need to know what is the most important thing to focus on first. For the sake of argument, we will also assume that we are working with organizations which definitely would choose one priority over another.

For most organizations, courtesy is unquestionably the number one priority. One of the interesting realities of customer support is that, barring utter and complete failure in any one of the other areas, as long as you are polite and considerate, people will tend to be patient with you and your organization. If someone makes a mistake or is unable to help you but is very apologetic and empathetic, you're likely to forgive him. Certainly there are limits to this, but it is very effective.

Second, and extremely close to courtesy, is response time. You can be sweet as a bunny covered with daisies, but if you never get around to actually talking to your customers they're not going to know that, and, more to the point, they'll think you don't know that they exist and are waiting for you to help them. Being certain to let customers know you will attend to them soon makes them feel satisfied that they are being attended to, even if you are not helping them at that moment.

Cost is very often the last thing on a customer's priority list.

Most customers are usually quite concerned with turnaround time, and it usually falls around third. Needless to say, customers want to have their needs fulfilled within a reasonable period of time. A customer may never get a chance to determine what your turnaround time is like, however, if she gets tired of waiting for you to indicate that you know she is there and intend to help her before she turns to dust. She is also unlikely to hang around waiting if staff are glaring at her and making rude remarks.

Convenience is a surprisingly powerful motivator but usually falls around fourth in the list. Customers are likely to come to your store first so long as you are reasonably competent in other customer-support priorities/categories. They will, however, definitely go out of their way to find a support organization or store that will be filled with happy, daisy-covered bunnies who hop over immediately to say hello and are always good about bagging your fish quickly and with a smile. (Wow – there's a mental image for you!) They will not, however, go to the place if it has rude, inattentive, uncaring staff, even if it is right next door.

Oddly enough, selection, whether tangible or otherwise, is often very low on the list. Certainly, if you literally don't have, and cannot get, what the customer needs, he cannot purchase it from you. Assuming, however, that you score high on all of the other priorities, the customer will most likely try your store first, and will often opt to wait for a special order through you before going to another vendor. Standard rules about happy bunnies, of course, still apply.

This brings us to cost. Surprise! Cost is very often the last thing on a customer's priority list. There is no question that people will pay extra to obtain more of any of the core components listed above. It is easy to come up with hundreds of examples, but certainly you can think of some yourself. Did you pay more to go to a restaurant with nicer food than McDonald's? More pleasant wait staff? Perhaps you went to a more expensive car wash because at the cheaper one the scary man with the big washing broom was making lewd remarks? The list could go on for pages.

So let's throw those items into a list in order of priority so we can analyze what we have:

Priority	*Core Component*
1	Courtesy
2	Response Time
3	Turnaround Time
4	Convenience
5	Selection (tangible/intangible)
6	Cost

How does this compare to our list of ratings, or core competencies, with The Pet Stop? Does it line up exactly, or are there, perhaps, a few differences? How significant are they? Why is that so, and why do you think The Pet Stop is a successful business? If a priority is high, but the Pet Stop rated low on the scale, why are they still making money? Think about these questions for a while, then read on.

Now I would like you to take out the list you created after you read my last article [*;login:*, December 1999]. Look at the places you visit and carefully examine each of the items you listed as priorities for why you chose that service or product over another. Select from the list of core components of customer support the priority that most closely fits each of the priorities that you listed on your sheet. Now compare them against our priority list and see how close you came. Did your list of priorities match the list above? Was it different? If so, why?

It is likely that many of you are looking at the information you just fleshed out and are realizing that it is somewhat, or even completely, different from the priority list above. You may even be thinking that I am completely wrong and have absolutely no idea what I'm talking about.

For those of you out there who are in this position, I say "Good!" In fact I say, "Perfect! Could you please stand up and wave to the remainder of the people in the audience." To everyone who is still sitting, I will now say that the people standing are a fantastic illustration for the next article in the series. My next article will focus on a modifier to this one, which will throw a rather heavy wrench into the works because it will bring to light that every customer is different!

As an exercise, think more about my local pet store and its list of strengths and weaknesses. Think more about why certain customers choose it, and why certain customers would not. If you were one of the people whose priorities were not in the same order as the list above, try to determine whether you would or would not go, based upon the description that we used. Try to identify why you have said "yes" or "no," and look for reasons in your list of priorities. If you find many inconsistencies, attempt to explain them using other priorities and how they compensate.

If you happen to visit a pet store for inspiration, beware of fuzzy bunnies bearing baggies of fish.

application note

by Eljakim Schrijvers

Eljakim Schrijvers is an independent consultant residing in central Holland who specializes in database applications for the Web. He also coaches the Dutch Computing Olympiad and is an avid underwater hockey player.

<info@eljakim.nl>

[Editor's note: Application Notes are short articles that solve specific problems that might arise at readers' sites. Feel free to submit short articles like this to <login@usenix.org>.]

Problem

I have a private network that is connected to the outside world via a firewall. The private systems are invisible to the outside world. The firewall is running Linux. On the inside is a Windows NT server that is serving several domains. I had all the IP masquerading up and running and everything was running smoothly until I wanted to serve one of the domains from the firewall itself, and another domain from another Linux server. So I had to stop blindly forwarding the IP packages for port :80 to the private server. I also couldn't just redirect my outside clients to a different IP address based on the domain name, because the private servers are invisible to the outside world.

Solution

STEP #1

I figured out the ProxyPass option of Apache. This option lets your server serve Web pages from a different domain as if you had the entire domain mirrored. You have to provide the virtual path under which the other domain will be a server. For instance, proxypass /foo/ http://www.foo.com/ will serve *http://www.foo.com/bar* when a client requests *http://yourhost/foo/bar*. Using / for virtual path will serve everything from the other server. Note that this is something other then redirecting, since the client will not know where you're getting the data from. It will think you are serving all the requests.

I edited the httpd.conf file and added the following lines:

```
    <NameVirtualHost EXTERNALIP>

# Pass all requests for www.myfirstdomain.com on to
# the server that has ip address PRIVATEIP1
    <VirtualHost EXTERNALIP>
        ServerName www.myfirstdomain.com
        ProxyPass / http://PRIVATEIP1/
    </VirtualHost>

# Pass all requests for www.myseconddomain.com on
# to the server that has ip address PRIVATEIP2
    <VirtualHost EXTERNALIP>
        ServerName www.myseconddomain.com
        ProxyPass / http://PRIVATEIP2/
    </VirtualHost>

# Handle all requests for www.mythirddomain.com from
# directory /www/mythirddomain/
    <VirtualHost EXTERNALIP>
        ServerName www.mythirddomain.com
        DocumentRoot /www/mythirddomain/
    </VirtualHost>
```

Note that even though IP addresses PRIVATEIP1 and PRIVATEIP2 are not visible to the outside world, the Web pages are served by the correct server.

If every internal server would serve requests for only one domain, you would be done by now. However, the transition that Apache does loses the original domain name. Apache requests a document from the internal server and approaches the server by its

IP address. In order to serve multiple domains from one internal server, I had to find a way to keep the domain name from getting lost during the transition.

STEP #2

I set up the nameserver of the firewall (named) to resolve *www.myfirstdomain.com* and *www.myseconddomain.com.* It resolves the names to the private IP address. Because the outside world does not use my firewall as a DNS server, this does not cause any problems. I could now tell Apache to pass all requests for *www.myfirstdomain.com* on to the same domain. Since the domain name resolves to the private server, everything now works fine.

Even though the new httpd.conf file looks rather trivial, this method works perfectly.

```
    <NameVirtualHost EXTERNALIP>

# Pass all requests for www.myfirstdomain.com on to
# the server that has ip address PRIVATEIP1
    <VirtualHost EXTERNALIP>
      ServerName www.myfirstdomain.com
      ProxyPass / http://www.myfirstdomain.com/
    </VirtualHost>

# Pass all requests for www.myseconddomain.com on
# to the server that has ip address PRIVATEIP2
    <VirtualHost EXTERNALIP>
      ServerName www.myseconddomain.com
      ProxyPass / http://www.myseconddomain.com/
    </VirtualHost>

# Handle all requests for www.mythirddomain.com from
# directory /www/mythirddomain/
    <VirtualHost EXTERNALIP>
      ServerName www.mythirddomain.com
      DocumentRoot /www/mythirddomain/
    </VirtualHost>
```

dot-coms' newest secret weapon doesn't have name

by Lenny Liebmann

Lenny Liebmann is a consultant and writer specializing in the application of computing and communications technologies to real-world business challenges. His work appears in *Computerworld*, *Internet Week*, *Network Magazine*, and other leading IT publications.

<LL@exit109.com>

Editor's Note: I am pleased to reprint this article by permission of the quarter-million circulation ComputerWorld *magazine, from whose January 31, 2000, issue it came. This article is important because it is among the first to recognize in a large public forum that a huge disparity exists among the skills of system administrators and that those who are at the high end of the scale can make dramatic improvements to business's bottom line. He references the interesting term "über-geek" and uses the term "holistic" in reference to senior administrators' abilities – I really liked that. It is articles like these that show that systems administration is finally being recognized as a true profession unto itself.* – RK

There's an absolutely critical new skill that most IT staffs sorely lack. It's a skill that clearly separates successful dot-coms from e-commerce wanna-bes. It's also a skill that doesn't really have a name yet. For now, let's just call it Web-application scalability engineering. It's part science and part art, and it's crucial to dealing with the new economics of e-commerce.

I know what you're thinking: "We have Java programmers. We have UNIX systems administrators. They know how to build systems that scale." But do they? Or do they just know how to throw hardware at a problem?

I've seen dot-coms achieve 160 times the capacity of a corporate Web site with a budget one-fifth as big. How do they do it? They're sure not gonna tell you, because it's at the core of their competitive advantage.

But what I can tell you is the dot-coms have some smart technical people on their teams who take a very cross-disciplinary approach to achieving scalability. These people understand networking issues like Internet service provider peering (how their providers interconnected with other ISPs) and payload-to-header ratios (a factor that determines how efficiently they use their available bandwidth). They understand systems issues like processor utilization and caching. They understand software engineering issues like database connection pooling (which reduces the strain on back-end resources) and intelligent agents. Their holistic view of how these components affect a customer's ability to get a fast response when they click on a Web-page button makes them formidable competitors.

These new IT alchemists know how to stretch a budget, too. They have zero tolerance for software licensing schemes that penalize them for success. They'd rather write a database from scratch than shell out six figures for the privilege of running a vendor's solution across 48 commodity Intel servers. They understand that scale isn't just about MIPS and megabits per second. It's about dollars and cents.

You see, you can't achieve scale when you're still taking a stovepipe approach to the problem, divvying up infrastructure responsibilities and application developers. The new breed of technosavant is a polymath who understands that to achieve end-to-end performance you need to end-to-end perspective. That end-to-end mentality is something to which most IT shops still only give lip service. But the über-geeks who ride shotgun on the Web's top sites have it scoped. If you don't think your company's success hinges on Web-application scalability engineering, just look at outfits like DoubleClick, Amazon.com, and Motley Fool. They've been able to rise as far and as fast as they have because they've cracked the code of scalability. And they did it before the cash started pouring in.

The Web operates under a set of technical and economic principles vastly different from the world of client/server application development. At the heart of that difference is scale. Once your site starts attracting heavy traffic, scalability is what makes the difference between revenue and ridicule. Recognize it as a distinct discipline. Cultivate it among the members of your staff with the right competencies and disposition. Or prepare yourself to become the next Web scandal. Those are about your only choices in the megahit per second world of the Web.

"this company doesn't . . ."

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

<scj@transmeta.com>

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and trouble-shooter for technical companies.

<dustywhite@earthlink.net>

An employee comes into your office and says something like, "This company doesn't appreciate me. I could get better offers elsewhere." As a manager, you have just been given a gift.

In the vast number of cases, when employees quit, the first thing they explicitly say to a manager about their job satisfaction is "I quit!" If the employee is still talking, there is great hope that the situation can be improved. In fact, one of the rationales for having periodic performance reviews is precisely to get unhappy employees to talk about their concerns before they get too frustrated and quit.

So why is such a conversation a gift? Realize that the employee is taking a risk by expressing his negative reaction to the company. By being willing to talk to you, he are showing a degree of trust in you. But it's necessary to proceed carefully to avoid making the situation worse.

When employees say, "This company doesn't . . . ," what do they mean? Companies don't do things, people do. Companies may have procedures and a thick rule book, but people often don't follow it. Someone unhappy about discrimination, for example, may be well aware that the company policy forbids it, but this doesn't keep some people from doing it anyway.

In an earlier column, we encouraged you to use the question "how?" to understand the process behind an undesirable outcome. And we will encourage "how questions" here as well, but with care.

First, we need to acknowledge the trust that the employee is showing by having this discussion at all, and deal with her emotional state, which may be fearful, angry, injured, resentful, guilty, anxious, etc. And you probably are seeing only a small piece of what she is feeling. If you don't acknowledge employees' feelings but instead plunge right into problem-solving, you will leave them dissatisfied, and may well appear defensive or insensitive to them.

So the first thing to do is to listen. Ask questions that chunk them up (see our column in the February 2000 issue) – don't let them get caught in a list of dozens of petty incidents, but ask them, "How has this affected your ability to do your job?" "how would you rather have been treated?" and other questions to give them more context. The facts really are less important at this stage than the employee's feeling that something has happened. So resist the urge to elicit chapter and verse about who did what when, and stay focused on the employee's impressions. Often, if you listen sympathetically and nonjudgmentally, and gently chunk up, the emotions will subside, and the employees will themselves suggest an excellent approach to solving the problem.

At this stage, you can take the lead, and start asking "how" questions that chunk down bit. Gather those details you need to, and suggest different ways of dealing with specific situations. In some cases, you may need to go to your own manager to make the right things happen. In some cases, you may not feel empowered to fix the problem; this will be the theme of a future column.

Now wait! Don't just rush in and offer suggestions too quickly! One of the most common mistakes managers make is assuming that the "presenting problem" or issue is the only one. They then get out their "box of Band-Aids" and attempt to do a quick fix. Before making any suggestions about solutions, be sure to chunk down and get any other issues that may have not come up. For instance, Joe may be upset about the way co-worker talks to him and orders him around. That may be only the straw that broke the camel's back, and there are really a number of little issues that prompted the blow-up.

You'll be doing your employee and yourself a favor if you take the time to find out all the issues. By keeping open, encouraging your employee to talk, using phrases such as "tell me more" and "what else?" you get the larger picture of the problem and possibly pattern that needs to be addressed, rather than just one issue.

But above all, realize that having this discussion with your employee is a gift. It is far superior to "I quit!"

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

A number of good books came my way in December and January (and several bad ones, but I won't mention those). I also flew around a decent amount, so I was able to get enough reading done.

Let me start and end with books that aren't really about computing at all.

Databases

All of us are aware that gazillions of bits about all of us are maintained in the memories of doctors, schools and colleges, credit card issuers, etc. And we assume that these bits are more or less inviolate. But they're not.

A few years ago I reviewed a book on data mining. Now I've read Simson Garfinkel's opus. Perhaps I should write "polemic." Garfinkel is out to warn us of the death of our notions of privacy: looked at a pornographic site? defaulted on a loan? been cited for DUI? late with the rent? suffering from some ailment best kept from your employer? – all such material is accessible to the assiduous data miner.

I don't want to spoil your fun, so I'm not going to detail what Garfinkel says. Suffice it for me to say that I read *Database Nation* straight through in one evening. Garfinkel's semijournalistic style is terrific and his content is terrifying.

A definite must-read.

NFS

Another really good book is Callaghan's on NFS. In many ways it is the ideal tribute to the late Rich Stevens, whose work on TCP/IP is cited. Callaghan has been working on NFS at Sun for about 15 years. (Sun announced NFS toward the end of 1984.) He is also one of the authors of the NFSv3 Protocol Specification (RFC 1831).

It is just about a decade since Hal Stern wrote his book on NFS. The need for a more contemporary one has been great. Callaghan has succeeded in writing a volume that "tells it all" and in which you can actually find what you need.

BASH

I have to admit that I don't use BASH. I don't use ksh, either. I generally use csh and occasionally zsh. And, of course, I resort to sh on occasion.

But I decided to be open-minded with Tansley's book. And it is a genuinely solid introduction to BASH – the shell that's available to all UNIX and Linux systems that I've come in contact with.

Tansley has organized his book around five topics:

the shell commands and syntax
text-filtering tools
login environment and customization
basic shell programming
advanced techniques

I think it's more in tune with the beginning BASH user, or someone who knows only a few things (like grep, awk, and sort) than the advanced programmer in search of insight. Of great utility are Tansley's occasional "If it's Linux then . . . " boxes.

But it is a good book (even if Brian Kernighan's and Peter Weinberger's names are misspelled on p. 97).

Java

I thought I'd never read another Java book, but I read most of Haggar's and

BOOKS REVIEWED IN THIS COLUMN

DATABASE NATION: THE DEATH OF PRIVACY IN THE 21ST CENTURY
SIMSON GARFINKEL
Sebastopol, CA: O'Reilly, 2000. Pp. 312.
ISBN 1-56592-653-6.

NFS ILLUSTRATED
BRENT CALLAGHAN
Reading, MA: Addison-Wesley, 2000. Pp. 511.
ISBN 0-201-32570-5.

LINUX & UNIX SHELL PROGRAMMING
DAVID TANSLEY
Reading, MA: Addison-Wesley, 2000. Pp. 504.
ISBN 0-201-67472-6.

PRACTICAL JAVA PROGRAMMING LANGUAGE GUIDE
PETER HAGGAR
Reading, MA: Addison-Wesley, 2000. Pp. 279.
ISBN 0-201-61646-7.

CONFIGURING CISCO VOICE OVER IP
ELLIOTT LEWIS ET AL.
Rockland, MA: Syngress, 2000. Pp. 512.
ISBN 1-928994-03-2.

THE CLUETRAIN MANIFESTO
RICK LEVINE ET AL.
Cambridge, MA: Perseus Books, 2000. Pp. 190.
ISBN 0-7382-0244-4.

ELEMENTS OF SEMIOTICS
DAVID LIDOV
New York: St. Martin's Press, 1999. Pp. 288.
ISBN 0-312-21413-8.

found it informative. This is truly a "practical guide," so those of you who aren't language mavens can now avoid *The Java Language Specification*, which I think is a wonderful work.

Haggar has a forthright, practical style, and *Practical Java* is exactly that.

A Disappointment

Having read Uyless Black's book last year, I was wary of *Configuring CISCO Voice over IP*. If you want the facts of a Cisco installation, here are diagrams of, for example, VIC-2E/M and VIC-2FXS. I found the two appendices on IPv6 the most interesting part of the book. For understanding, I'd go for Black.

And Another

I'm at a loss as to what to say about the *Cluetrain Manifesto*. First of all, I know several of the authors; second, I was looking forward to the book. Well, the basic thrust of the Manifesto is that the Internet has changed and will continue to change the very nature of markets. Secondarily, there is the insight that markets are conversations.

"Markets are conversations" is thesis #1 of the 95 the authors list at the front of their book. Of course, there are 95 because Martin Luther had 95 Theses which he nailed up on the door of the castle church in Wittenberg on October 31, 1517. Luther's 95 Theses concerned the Church's practice of selling indulgences. These theses are literally all over the map. The seven essays that follow are also diffuse and flabby.

Chris Locke's chapter is full of some of the worst rhetoric I've ever read. "Dark and stormy night" comes nowhere near:

> Dig deeper. Down to the sites that never entertained the hope of Buck One. They owe nobody anything. Not advertisers, not VC producers, not you. Put your ear to those tracks and listen to what's coming like a freight train. What you'll hear is the sound of passion unhinged. (p. 35)

Weinberger's chapter asks, "What's the Web for?" He concludes that it is "to build a new world" (p. 45).

Rick Levine tells me, "Wise companies will learn how to enter the conversation"; Doc Searls tells corporate structures, "We, the market . . . want to speak with your business in a human voice." Etc., etc. "Hyperlinked organizations," "change of historic proportions," "the revolution . . . is already well underway." As is the torrent of overwritten bombast.

I think the authors' points are valid. This would have made a terrific short article.

Another Goodie

Our species manipulates signs. Computing, graphic art, speech, etc. are all rooted in signification. Late last year I got a copy of a book by David Lidov. I knew him as a professor of music at York University in Toronto. I even own a CD of his stuff.

But this book has only a bit about music, and a great deal about everything else.

Lidov avoids philosophy while considering how things mean and how the "artifacts of mental life" are integrated in each of us.

Not about computing, but can one find meaning without inquiring what "to mean" and "to refer" mean or refer to?

standards reports

The POSIX Bookshelf

by David Blackwood
Standards Reports Editor
<dave@usenix.org>

Our Standards Reports Editor, David Blackwood, welcomes dialogue between this column and you, the readers. Please send your comments to <dave@usenix.org>

This article is the first in a series that will provide you with a list of published sources of information about the POSIX family of standards. This month we begin with the standards themselves.

The IEEE Portable Application Standards Committee (PASC) develops the POSIX standards. In accordance with a synchronization plan adopted by the IEEE and ISO/IEC JTC1, many of the POSIX standards become International Standards shortly after their adoption by the IEEE. Therefore, these standards are available in printed form from both IEEE and ISO, as well as from many national standards organizations. Approved standards can also be purchased from the IEEE in electronic (PDF) format, usually at a premium over the printed-copy cost. As well, the IEEE publishes Standards Interpretations for many of the standards. Contact the IEEE for further details or visit their Web site at *<http://standards.ieee.org/>*.

The current IEEE POSIX publications:

1003.0-1995 IEEE Guide to the POSIX® Open System Environment (OSE) (*identical to ISO/IEC TR 14252*). 288 pages. ISBN 1-55937-531-0. $98.00.

9945-1:1996 (ISO/IEC) [IEEE/ANSI Std 1003.1, 1996 Edition] Information Technology – Portable Operating System Interface (POSIX®) – Part 1: System Application: Program Interface (API) [C Language]. 784 pages. ISBN 1-55937-573-6. $143.00. *This edition incorporates extensions for realtime applications* (1003.1b-1993, 1003.1i-1995) *and threads* (1003.1c-1995).

1003.1d-1999 Information Technology – Portable Operating System Interface (POSIX®) – Part 1: System Application Program Interface (API) – Amendment x: Additional Realtime Extensions [C Language]. $57.00.

9945-2:1993 (ISO/IEC) [IEEE/ANSI Std 1003.2-1992 & IEEE/ANSI 1003.2a-1992] Information Technology – Portable Operating System Interface (POSIX®) – Part 2: Shell and Utilities. 1328 pages. ISBN 1-55937-406-3. $271.00. *Includes and is shipped with 1003.2d-1994.*

1003.2d-1994 IEEE Standard for Information Technology – Portable Operating System Interface (POSIX®) – Part 2: Shell and Utilities – Amendment 1: Batch Environment. 152 pages. ISBN 1-55937-512-4. $81.00.

14519:1999 (ISO/IEC) IEEE/ANSI Std 1003.5b-1999 Information Technology – POSIX® Ada Language Interfaces – Binding for System Application Program Interface (API) – Realtime Extensions. 548 pages. ISBN 0-7381-1570-3. $95.00.

1003.5-1999 Edition IEEE Standard for Information Technology – POSIX® Ada Language Interfaces – Part 1: Binding for System Application Program Interface (API). 890 pages. ISBN 0-7381-1539-8. $145.00. (*This edition incorporates IEEE Std 1003.5-1992, IEEE Std 1003.5b-1996, and IEEE Std 1003.5c-1998.*)

1003.9-1992 (R1998) IEEE Standard for Information Technology – POSIX® FORTRAN 77 Language Interfaces – Part 1: Binding for System Application Program Interface (API). 186 pages. ISBN 1-55937-230-3. $65.00.

1003.10-1995 IEEE Standard for Information Technology – POSIX® – Based Supercomputing Application Environment Profile. 80 pages. ISBN 1-55937-546-9. $71.00.

1003.13-1998 IEEE Standard for Information Technology – Standardized Application Environment Profile (AEP) – POSIX® Realtime Application Support. 186 pages. ISBN 0-7381-0178-8. $77.00.

1003.23-1998 IEEE Guide for Developing User Organization Open System Environment (OSE) Profiles. 120 Pages. ISBN 0-7381-1541-X. $60.00.

15068-2:1999 (ISO/IEC) [IEEE/ANSI Std 1387.2-1999] Information Technology – Portable Operating System Interface (POSIX) System Administration – Part 2: Software Administration. 280 pages. ISBN 0-7381-1568-1. $70.00.

1387.2-1995 IEEE Standard for Information Technology – Portable Operating Interface System – Part 2: Software Administration. 296 pages. ISBN 1-55937-537-X. $86.00.

1387.3-1996 IEEE Standard for Information Technology – Portable Operating System Interface (POSIX®) System – Part 3: User and Group Account Administration. 112 pages. ISBN 1-55937-866-2. $71.00.

13210:1999 (ISO/IEC) [IEEE Std 2003-1997] IEEE Standard for Information Technology – Requirements and Guidelines for Test Method Specifications and Test Method Implementations for Measuring Conformance to POSIX Standards. 96 pages. ISBN 0-7381-1797-8. $86.00.

2003-1997 IEEE Standard for Information Technology – Requirements and Guidelines for Test Method Specifications and Test Method Implementations for Measuring Conformance to POSIX Standards. 96 pages. ISBN 1-55937-895-6. $71.00.

2003.1-1992 IEEE Standard for Information Technology – Test Methods for Measuring Conformance to POSIX® – Part 1: System Interfaces. 456 pages. ISBN 1-55937-275-3. $114.00.

2003.2-1996 IEEE Standard for Information Technology – Test Methods for Measuring Conformance to POSIX® – Part 2: Shell and Utilities. 1424 pages. ISBN 1-55937-882-4. $119.00.

The following unapproved drafts of IEEE Standards Projects are also available for purchase, in printed form only. Since these are subject to change, the IEEE warns that use of the information in these drafts is at your own risk. Other drafts may also be available. Contact the IEEE at *<http://standards.ieee.org/>*.

P1003.1a, Draft 16, December 1998. System Application Program Interface (API) [C Language] – Amendment C (*additional system services*). $54.00.

P1003.1g, Draft 6.6, March 1998. Protocol Independent Interfaces (PII). $82.00.

P1003.1h, Draft 5, July 1999. POSIX Part 1: System API Extension – Services for Reliable, Available, and Serviceable Systems [C language] – Amendment (*fault tolerance*). $41.00.

P1003.1j, Draft 9, July 1999. Advanced Realtime Extensions [C Language]. $49.00. (*Current draft is draft 10.*)

P1003.1m, Draft 2, November 1998. Checkpoint/Restart Interface [C Language]. $38.00.

P1003.1q, Draft 5, July 1999 Tracing [C Language]. $44.00. (*Current draft is draft 7.*)

P1003.2b, Draft 12, June 1999. Shell & Utilities, Amendment 2 (*additional utilities*). $73.00.

P1003.14, Draft 10, February 1995. IEEE Standard for Information Technology – POSIX® Standardized Profile – POSIX Multiprocessor Application Environment Profile. $42.00. (*Withdrawn.*)

P1003.18, Draft 12, February 1995. POSIX Interactive Systems Application Environment Profile. $40.00. (*Withdrawn.*)

P1003.21, Draft 2.0, August 1998. Realtime Distributed System Communication LIS. $61.00. (*Current draft is V3.0.*)

Austin Group Status Update

by Andrew Josey

Andrew Josey is the Director, Server Platforms, for The Open Group in Reading, England, and the chair of the Austin Group.

<a.josey@opengroup.org>

The Austin Common Standards Revision Group (CSRG) is a joint technical working group established to consider the matter of a common revision of ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std 1003.2, and the appropriate parts of the Single UNIX Specification.

As of February 2000, two drafts have been produced, totaling some 2,700 pages of material. At the time of writing, preparation is well underway for Draft 2. Draft 3 integrates materials from the following base documents:

Networking Services Issue 5 Version 2
IEEE Std 1003.1d-1999
IEEE Std 1003.2d-1994
IEEE Draft Std P1003.1a
IEEE Draft Std P1003.2b
IEEE Draft Std P1003.1j

It is worth noting that this will be the first revision to the Commands and Utilities volume since 1992.

After production of Draft 3, there is a two-month review period before the plenary meeting in May. Comments should be sent to the review reflector and use the aardvark comment format (see <http://www.opengroup.org/austin/aardvark/format.html> for more information).

During the review period, a report on the changes needed for alignment with c99 will be produced. This can then also be discussed in plenary before Draft 4.

Draft 4 will be the feature-complete draft of the specification, and the new features over Draft 3 that are anticipated (subject to agreement of the plenary) are c99 alignment and 1003.1q.

The next plenary meeting will be held during the week commencing May 15th.

Summary slides giving the latest Austin Group status are available at *<http://www.opengroup.org/austin/materials/jan2000/>*. Slides from the January teleconference are available at *<http://www.opengroup.org/austin/materials/jan2000/teleconference/>*.

To participate in the Austin Group visit the Web site at *<http://www.opengroup.org/austin/>*.

Recent Publications from The Open Group

by Andrew Josey

<a.josey@opengroup.org>

Networking Services Issue 5.2

Abstract: The Networking Services (XNS) Issue 5.2 Technical Standard defines the industry-standard Open Systems interfaces to communication services. These comprise the Sockets Interface and the X/Open Transport Interface (XTI). The Sockets Interface (part 2 of the Standard) is now the main industry-standard interface. XTI (part 3 of this Standard) is now considered obsolete, so writers of new applications using the Internet Protocol Suite (IPS) are recommended to use Sockets rather than XTI. Where protocols for which there is no sockets support are in use, XTI is still recommended in preference to proprietary APIs. XNS 5.2 supersedes the previous XNS publication, XNS Issue 5 C523 (Feb 1997). The most important new feature is the inclusion of Internet Protocol version 6 (IPv6) functionality, in a manner that is aligned with the relevant IETF IPv6 standard (RFC 2553).

The Web version is available at *<http://www.opengroup.org/pubs/catalog/c808.htm>*.

Transport Provider Interface Version 2

Abstract: The Transport Provider Interface (TPI) Standard defines an interface for drivers that provide transport services in a networking environment. The TPI defines the set of messages and their formats that the driver must generate and process. This specification is firmly based on the TPI specification originally produced by UNIX International (UI).

The Web version is available at *<http://www.opengroup.org/pubs/catalog/c810.htm>*.

Data Link Provider Interface (DLPI) Version 2

Abstract: The Data Link Provider Interface (DLPI) Standard defines a STREAMS kernel-level instantiation of the ISO Data Link Service Definition DIS 8886 and Logical Link Control DIS 8802/2 (LLC). Where the two standards are not aligned, DIS 8886 prevails. The DLPI interface enables a data link service user to access and use any of a variety of conforming data link service providers without special knowledge of the provider's protocol. Specifically, the interface is intended to support X.25 LAPB, BX.25 level 2, SDL C, ISDN LAPD, Ethernet, CSMA/CD, FDDI, token ring, token bus, Bisync, Frame Relay, ATM, Fibre Channel, and HIPPI. This list will be added to as new protocols are deployed.

The Web version is available at *<http://www.opengroup.org/pubs/catalog/c811.htm>*.

The HTML versions of the above can be accessed without restriction. Members of the Open Group, including participants in the Austin Group, can use their Web ID and password to access the PDF versions.

USENIX news

The following two articles are reports from projects supported by USENIX grants. To borrow from common highway signs: "Your dues dollars at work."
The Editors

Tomorrow's Scientists and Urban Planners Create the City of the Future

by Noel N. Kriftcher

Dr. Kriftcher is Director of Polytechnic University's David Packard Center for Technology and Educational Alliances and former superintendent of high schools in Brooklyn and Staten Island, New York City.

<nkrift@duke.poly.edu>

Saturday, January 29, was a crisp, sunny winter day in New York City, absolutely perfect weather for this year's Metropolitan New York Regional Finals in the Future City Competition, held at Polytechnic University's MetroTech Campus in Brooklyn, New York. Conducted in conjunction with the American Society of Civil Engineers (ASCE), this event concluded four months of hard work for more than 100 seventh- and eighth-grade students drawn from New York City's five boroughs, Long Island, Westchester, and New Jersey, as well as for the teachers and mentor-engineers who guided their activities. For the overall team winner, the event served as a stepping stone to the national Future City finals, held in Washington, D.C., during National Engineers Week, in mid-February.

This is the third year that Polytechnic has conducted this competition, under the aegis of the David Packard Center for Technology and Educational Alliances. Encouraging the development of "active" learning experiences is a goal of the Center, as is the commitment to ensure equity of availability, opportunity, and access for women and underrepresented minorities in the use of computers, information-age technology, and the study of mathematics and science. Future City, therefore, is a natural bridge to prepare middle-school students for advanced study, one that effectively embodies one of the university's missions, to forge alliances to support pre-collegiate scientific study.

By participating, students develop computer skills, problem-solving skills, team-building techniques, practical mathematics and science applications, research skills, and presentation skills. Starting with their learning to use SimCity 2000 software, they conduct research into issues facing the urban environment, create graphs of their personalized vision of their ideal city, write essays that analyze and explain their concerns, build three-dimensional scale models that convey their view of the "City of the Future," and offer oral and visual presentations to expert panels of engineers and environmental scientists who volunteer as judges for the Regional Finals.

The New York City School System has developed "Applied Learning Performance Standards," which are based on the work of the Commission on Achieving Necessary Skills (SCANS) of the U.S. Secretary of Education. These define how closely each student's experience with Future City mirrors the five performance standards developed for Applied Learning:

- Problem solving;
- Communication tools and techniques;
- Information tools and techniques;

- Learning and self-management tools and techniques;
- Tools and techniques for working with others.[1]

Wisely, these performance standards for Applied Learning do not suggest that this be a separate subject. Rather, the standards seek to identify skills that should be interwoven within the context of different subjects. In the case of Future City, these standards serve to complement such necessary skill areas to be developed within the science curriculum as "scientific thinking; scientific tools and technologies; scientific communication; and scientific investigation."[2]

One student after another commented on how much "fun" he/she had in developing the models, each of which was limited to one moveable part in order to equalize expenses and provide a uniform format. There were bridges that moved, underground highways, solar panels, even a baseball/football stadium with a retractable dome that opened 21 minutes faster than the one in Toronto, we were proudly informed. Water supplies were protected, greenery abounded – for both aesthetic and environmentally sensitive reasons, we were assured – and multi-occupancy residential zones emerged. The scientists and engineers who served as judges, many of whom participated all three years we have sponsored this competition, commented repeatedly on the depth of the thinking that went into creating the models, and the quality of the presentations and "defenses" of the projects offered by these young people, who were drawn from diverse racial and social backgrounds and represented diverse educational experiences.

As a way of encouraging school completion, higher academic aspirations, college attendance, and the pursuit of studies (and careers) in science, mathematics, engineering, and technology, Polytechnic University offers to each member of the first-place team a $5,000 annual scholarship, contingent on being accepted and attending Polytechnic. The second-place team each receives an annual $2,000 scholarship, and the third-place team members, $1,000. In addition, the winning team receives an escorted, all-expenses-paid trip to the national finals in Washington, D.C. All participants receive medals from ASCE, teams scoring in first through fifth place receive trophies, and eight teams qualify for special awards, in categories such as "Most Creative Use of Materials," "Most Sustainable Transportation System," "Best Residential Zone," and "Most Environmentally Friendly." ASCE members solicit prizes to support these special awards from engineering, financial, and government concerns.

Special recognition must be given to the USENIX Association for its generous award to support Polytechnic's sponsorship of this activity. Dr. Helene Dunkelblau joined my staff this year as a consultant, for the express purpose of coordinating the many tasks that go into this event – recruiting schools, mentor-engineers, and judges; volunteers for the event itself; Polytechnic students who work with teams and assist with logistics; and the large planning group that is representative of both the university and ASCE. The USENIX grant supported the consultant's salary, student stipends, refreshments for students, volunteers and judges, rental of tables and chairs, and media activities to ensure that there would be a photographic record of the event. And the university supplemented this support by generously providing free space and expertise, such as that provided by the Media Office, which coordinated contacts with newspapers and television networks to secure positive attention for the students who entered this event.

Although this is a worthwhile, replicable project that serves a very important purpose in encouraging scientific study, an institution attempting to sponsor this activity should note certain caveats:

1. It is imperative that you have the complete cooperation of the local school system, and preferably some private and parochial schools as well as the public schools. It will be necessary to "sell" the project and enlist support from the district administration in order that those teachers who volunteer as "coaches" because they believe in the project not be allowed to become abandoned and isolated. The hours we have to devote, particularly as the date of the event draws closer, can reduce the strongest among us to tears unless we receive encouragement and support from our school's administration. Moreover, since it is permissible for a school to be represented by more than one team, this should be encouraged, both because of the positive peer pressure that will develop and because of the potential for meaningful curriculum reform as a result of shared-curriculum–based goal-setting.
2. Proximity to practicing engineers gives some schools an advantage over others in securing assistance. One school, fortuitously, had a teacher on staff who had been trained as an engineer, so he paired with a colleague who served as the team's teacher-coach. Email can help to ensure equal and continuing attention, even though the engineer may not be able to visit the school.
3. Someone has to be the taskmaster who is attentive to the deadlines for assignments. The program has grown from nine schools in 1998, to 17 in 1999, to 31 in 2000, but even this year there were still more than ten teams that did not complete various tasks in time for the finals.
4. Development of a functioning central planning team that communicates regularly and meets occasionally face-to-face is essential if the myriad tasks that go into conducting the event are not to become overwhelming in the final

days before the actual event. Responsibilities must be shared and clearly delineated. Who will maintain contact with each school and be responsible for status reports? Who will be the media contact? Who will create the program? Who will arrange for custodial services? Security? Video support? Breakfast and lunch? And who will coordinate all of these duties and ensure that tasks get completed?

5. During the week after the event, planning for next year should begin, starting with a debriefing to determine what needs to be changed and who should be thanked for special services rendered. Of course, letters that acknowledge the winning teams and any scholarships awards they have won must be disseminated, and acknowledgments should also be sent to the principals, school boards, and other important supporters of the schools' entries.

Students are afforded many opportunities to learn through television, computers, and libraries, but not every student is afforded an equal opportunity to participate in these experiences. Future City tries to democratize this exposure by offering an organized approach to applied scientific and social learning, under the guidance of responsible adults working together. Add to the mix a university working closely with a professional organization, and the possibilities for success are endless.

Polytechnic University is most pleased to take the lead in supporting learning in the precollegiate community and in finding other opportunities for young people to enjoy learning while working with adults who are willing to share, to guide, and to encourage. Through the leadership of the Packard Center, the university's walls have been extended outward to foster the integration of community organizations, teachers who wish to improve their skills, and students interested in experiencing academic enrichment. Under the banner of Polytechnic's Center for Youth in Engineering and Science, the "YES Center," secondary-school students have become a constant presence on campus, as they study in advanced credit-bearing courses both at Polytechnic and in their own high schools, and as they participate in a full range of activities that are offered during the summer as well as throughout the year. For example, students apply to participate in the YES Center's highly competitive Summer Research Program, which fosters the acquisition of research skills that prepare students for entrance into the Intel Science Talent Search and the International Science and Engineering Fair, through a qualifying round that is held at the university.[3] By encouraging the pursuit of rigorous academic courses and challenges, we offer the best chance for students to begin traveling down the road that leads to college success.[4]

Future City is but one of the programs we offer that bring young people together with mentors to make learning fun, raise their aspirations, and direct them to goal-oriented activities that synthesize academic knowledge with performance.[5] It is Polytechnic's commitment to the education of students well before they are eligible for university study that offers the best promise for the type of collaborative experience that builds a continuum of excellence.

RESOURCES

[1] National Center on Education and the Economy and the University of Pittsburgh, 1997, "Performance Standards, High School."

[2] The University of the State of New York: The State Education Department, 1996, "Learning Standards for Mathematics, Science, and Technology."

[3] The New York City Mathematics, Science, and Engineering Fair is a collaborative venture begun in 1998 among Polytechnic University, the New York City Board of Education, the New York

Academy of Science, and New York City Technical College of the City University of New York.

[4] United States Department of Education, 1997, "Getting Ready for College Early."

[5] United States Department of Education, 1998, "Yes, You Can: Establishing Mentoring Programs to Prepare Youth for College."

Software Patent Institute Report

by Roland J. Cole

Executive Director, Software Patent Institute

<cole@spi.org>

During 1999, the USENIX Association made a grant to the Software Patent Institute of $55,000, to be used by SPI to add materials to the SPI database. The Software Patent Institute is a nonprofit effort sponsored by a number of organizations concerned that decisions about patents for software-related inventions be based on the best possible information. One of SPI's major activities has been to develop and maintain a database of "software prior art" – that is, descriptions of software technologies from textbooks, manuals, technical reports, and journal articles. Because material from about 1990 on could be found in various online databases, SPI concentrated on materials not readily available, especially from the 1980s, 1970s, and 1960s.

Thanks to the USENIX grant, we were able to recreate the system SPI had developed for adding documents to its database. Thanks to increased experience on our part and to lowered prices and increased performance of computer hardware and software, the new system is much faster and more efficient than the SPI system of a few years ago. As we discuss in more detail below, we are now completing the processing of technical reports at a rate of sometimes more than 200 per month.

We ended the period of the grant with the database already expanded significantly in scope (thanks to the bibliographies already loaded) and expanding rapidly in size and institutional variety, thanks to all the technical reports. Continuing at this pace through the end of 1999, with no additional resources from USENIX, we have more than doubled the number of documents that were in the database at the end of 1998. We are tremendously grateful and are sure this collaboration between USENIX and SPI has extended the value of the database to all concerned.

WEB SITE
<http://www.usenix.org>

MEMBERSHIP
Telephone: 510 528 8649
Email: *<office@usenix.org>*

PUBLICATIONS
Jane-Ellen Long
Telephone: 510 528 8649
Email: *<jel@usenix.org>*

The SPI Database Development Process

In brief, our process is as follows. We first obtain either possession of a document of interest or copyright permission if it is required. Most of the documents come to us in paper form, although we receive some in electronic form. Many do not require copyright permission, as they are either government documents or published without a copyright notice prior to 1989 (usually by an academic or nonprofit institution). When we do need copyright permission, it can take a while. Almost every copyright holder has been willing to grant no-cost permission to our nonprofit effort. However, it often takes weeks or months to track them down, explain our proposal, and hear back from them. Alternatively, when we have copyright permission but do not have the actual document, obtaining the document can take a while. These are older materials, often out of print, and libraries are sometimes reluctant to loan them out (and often do not allow us to unbind and rebind them, since they have been rebound enough times before that the gutter margin is too small for yet another attempt).

Once we have both permission and physical possession of the documents, we then scan paper documents with a sheet-fed scanner. Then we use a software program (Caere Omnipage) that converts the image into text (albeit with varying degrees of accuracy) often called "OCR" (for optical character recognition, although the software uses language analysis as well as character analysis). If the document was difficult to OCR, we do our initial corrections in Omnipage, a process we call "proofing." The goal is not absolute correspondence to the original (that would take far too much time and other resources), but "good enough" correspondence that the search engine will work acceptably. Our goal is successful searching, with the idea that our database functions more or less as an extended "card catalog" rather than as a document delivery system, recognizing that our documents are not readily available for electronic searching unless we make them so. To use our resources most effectively, we omit nontextual material such as mathematical equations, segments of program code, figures, diagrams, and pictures, although we try to preserve the captions of such items.

If the document arrives in electronic form, we can obviously skip the scanning, OCRing, and proofing stages. We replace them with a stage we call "conversion," since we receive a variety of electronic formats and need to end up with ASCII text.

Our next step is to finish whatever "correction" of the document we feel is necessary, plus add the HTML-like (hypertext markup language) codes we use to identify headings, captions, and the like. We are trying to preserve the structure of the document – chapters, sections, and paragraphs – and to identify page breaks. We use Microsoft Word for this process and call this activity "formatting." Both paper documents and electronic documents go through this stage.

We then add a front end to the document which contains the citation material, such as author, publisher, and so forth.

Then we run the documents through a program written by SPI to check our formatting, a process we call "cleancheck." This program checks for mismatched tags, incorrect syntax, and the like. During the USENIX grant, we were able to convert this to an email process – our document specialist emails the supposedly formatted document to the SPI server in Michigan from her computer in Kansas City, and receives a return email (usually within seconds) that tells her the document is OK or lists the errors needing correction.

Next, this same program processes correctly formatted documents into records for the database, intelligently dividing documents into segments of approximately 10,000 bytes while keeping paragraphs, pages, and sections intact as much as possible.

The database program then takes the resulting collection of records and loads them into the database. We lump these last two steps together as the "loading stage."

As we said in our application for the grant, we had developed and honed this generic process from 1992 through 1997, but financial support had dropped to a level that enabled us to keep the database online but not to continue this process, let alone improve it. We processed approximately ten documents during the December 1997 through September 1998 period.

Thanks to the USENIX grant, we were able to start moving some 18,400 documents through our process during the December 1998 through November 1999 period. Along the way, we were able to restart our process and tune it to take advantage of major decreases in price and increases in performance in hardware and software between 1997 and 1999.

Before and After Process Improvements

Between December of 1998 (when the grant started) and November of 1999 (the end of this report period), SPI accomplished the following:

1. Upgraded hardware and software to the "state of the art" for standalone systems.
2. Redesigned the process to allow for steps to be separated and assigned to different people in different locations.
3. The net effect of 1 and 2 was to move from 10 pages per day to some 200 pages per day finished, with intermediate progress of several thousand

pages per day of scanning and OCRing.
4. Catalogued and started processing the 18,400 documents already in our possession, including some 36 boxes of IBM material obtained from the University of Michigan and some 110 boxes of technical reports obtained from Yale University.
5. The material added during this period includes the entire series of computer bibliographies complied by ACM from 1960 through 1994, plus technical reports from several dozen institutions, thus greatly expanding the scope of the database in both subject matter and institutional source.

Report of Documents in Process

The table presented below shows the active list in more detail. It shows the great variety in the institutional sources of these documents (they come from sources all over the U.S. and in Europe) and how many of them have already been scanned and OCRed, although not yet formatted.

The first number is the count, the second number is the stage of the process those documents are in. (Please note: all counts are approximate.) Also, we have continued processing since this table was prepared, in part because USENIX has agreed to renew its grant for 2000. In particular, we have scanned several thousand additional documents and loaded several hundred more.

1 = permission to be asked
2 = awaiting permission
3 = awaiting documents
4 = to be scanned
5 = scanned
6 = OCRed
7 = being proofed
8 = proofed
9 = received electronically
10 = converted
11 = being formatted
12 = formatted
13 = being cleanchecked
14 = passed cleancheck
15 = loaded

Document Source – count @ status

ACM Bibliographies – 9@15, 1@7
ACM Guides to Computing Literature – 14@15
Bolt, Beranek & Newman – 12@5
C.I.T. – 109@1, 61@5
Centrum voor Wiskunde en Informatica – 338@1, 105@2
Carnegie-Mellon – 70@1, 210@4
Columbia – 15@5
Cornell – 63@5, 5@15
Docs. by or regarding Smarandache – 55@4, 1@15
Docs. re: MIDAC – 9@4
Docs. re: IBM797 – 7@4
Docs. re: IBMAPL – 1@4
Docs. re: ORDVAC – 1@4
Ecole Polytechnique – 24@1
Harvard – 27@5
IBM Red Books – 54@4, 1@15
IBM Research Reports – 750@4
IBM System/370 Principles of Operation – 1@11
IBM Technical Disclosure Bulletins – 10@15
IEEE Computer – 29@15
IEEE Software – 6@15
Indiana University – 68@5
INRIA – 930@4
Iowa State – 16@4, 37@5
Israel Institute of Technology – 92@1
M.I.T. – 143@4, 152@5, 3@15
M.I.T. Dissertations – 50@1
McMaster University – 66@1
Michigan State – 27@5
Miscellaneous – 300@4
NASA – 295@4, 3@15
Newcastle Upon Tyne – 93@1
North Carolina State University – 24@5, 5@15
NYU – 24@4, 113@5
Old Dominion – 8@5
Ph.D. Dissertations – 11@8, 2@15
Princeton – 91@5
Purdue – 95@5, 5@15
RAND Abstracts – 3000@3
Rice University – 18@5
Royal Institute – 204@4, 7@5
Rutgers – 104@1, 218@4
Southern Methodist – 19@5
SRI – 38@5
Stanford – 310@4, 191@5, 8@15
Technical University of Munich – 710@2
Thinking Machines – 22@1
Turkey Biblio. – 21@1
U. of Alberta – 64@1
U. of Amsterdam – 302@1
U. of Arizona – 109@5
U. of British Columbia – 37@1
U. of California at Berkeley – 7@5
U. of California at Irvine – 11@5
U. of California at Santa Barbara – 18@4
U. of Central Florida – 33@5
U. of Colorado – 16@4, 33@5, 10@15
U. of Denver – 11@5
U. of Edinburgh – 132@1
U. of Helsinki – 15@1
U. of Illinois – 208@1, 950@4, 240@5
U. of Iowa – 14@4
U. of Kentucky – 46@5
U. of Kyoto – 28@1
U. of Maryland – 307@1, 95@4, 87@5, 4@15
U. of Minnesota – 128@5, 5@13
U. of New York at Buffalo – 23@5, 5@15
U. of Oregon – 1@15
U. of Paris – 28@1
U. of Pennsylvania – 10@4
U. of Pittsburgh – 11@4, 46@5, 4@13
U. of Rochester – 22@5, 4@13
U. of Saskatchewan – 160@1
U. of South Wales – 12@1
U. of Southern California – 12@4, 135@5
U. of Tennessee – 10@5, 5@15
U. of Texas at Austin – 75@4
U. of Toronto – 180@4
U. of Washington – 12@4, 75@5
U. of Waterloo – 303@1
U. of Wisconsin – 487@4
UCLA – 40@4, 20@5, 2@13
USENIX articles (from bibliography) 3808@1, 39@2, 23@3
USENIX Bibliography (from Web site) 1@15
V.P.I. – 14@5
Wang Institute – 21@1
Wayne State – 58@5
Weizmann Institute – 169@1
Xerox – 10@8, 5@15

Xerox Technical Reports – 28@4, 7@13, 10@15
Yale – 135@4

Totals by Status as of 9/30/99
(Note that items in any one stage have already gone through the previous stages – all those loaded have been formatted, for instance.)

1 = permission to be asked – 6533
2 = awaiting permission – 854
3 = awaiting documents – 3023
4 = to be scanned – 5610
5 = scanned – 2174
6 = OCRed – 0
7 = being proofed – 1
8 = proofed – 21
9 = received electronically – 0
10 = converted – 0
11 = being formatted – 1
12 = formatted – 0
13 = being cleanchecked – 22
14 = passed cleancheck – 0
15 = loaded – 142

Total Processed at Least in Part from December 1998 Through September 1999
18,431

With the renewal of the USENIX grant for 2000, we expect to increase the speed of the process once again, especially with major steps forward in hardware speed, and to at least double, if not quadruple, the total number of documents in the database during the year. The database is available without charge from SPI at <*http://www.spi.org*>, and we encourage everyone who is interested to explore what we are creating.

20 Years Ago in UNIX

by Peter H. Salus
USENIX Historian
<peter@eng.us.uu.net>

If you have your System III or PWB documentation handy, you might look at the title page. You'll find Ted Dolotta's name prominently displayed.

Ted was at Princeton, Bell Labs, and INTERACTIVE Systems in the '70s and '80s, and he just retired from Softbank, where he was vice president. He went to INTERACTIVE after a genuine courtship:

> One of the lures that Peter Weiner (who was a colleague of mine when we were both on the Princeton faculty) employed to lure me to INTERACTIVE was to give me his California UNIX license plates, which I recently sold via auction to John Mashey for $6,000, giving the money to the John Lions Scholarship Fund (and the charitable tax deduction to John Mashey).

Ted is a reader of these articles, and after my query about System IV he sent me the following:

> Hi, Peter,
>
> I noted your query in *;login:* regarding System IV UNIX at Bell Labs.
>
> As best I remember, there was indeed a System IV, but it was never offered for licensing via Western Electric. I no longer recall the exact reason why, but I think that it was not a major step beyond System III.
>
> One of the major improvements was a full set of application manuals in two volumes, some 1,000 pages (sed, troff, yacc, etc., etc. – some 50 or more manuals) which I edited for consistency and coherence and cross-references, etc.
>
> This document set was distributed internally with System IV, and externally with System V license. I left Bell Labs to join INTERACTIVE Systems right after this 2-volume set went to the printer, but before it was actually distributed.
>
> Also, System V was already in the planning & early development stages, so someone (Western Electric?) must have opted to wait for System V for licensing purposes and skipped licensing System IV.

Things lead to one another. Discussing the System III documentation with Ted, I elicited the following:

> Until we started doing System III at Bell Labs, all UNIX manuals were standard, letter-size (8.5" x 11"), loose-leaf manuals, and every update had only *new* pages and an update sheet: "Remove pages . . . , insert pages . . . ").
>
> When we were doing System III, I got to thinking: we were about to send out N thousand update packages to N thousand users within the Bell Systems (never mind licensees outside the Bell System), and we'd be propagating the fiction that *every one* of these highly-paid individuals would take the time to punctiliously update his/her copy of the manual. Right. We knew that, for practical purposes, every copy of the manual was unique – no two were alike. A genuine support nightmare. . . . And in terms of wasted technical talent hours spent updating thousands of manuals, mind-bogglingly expensive.
>
> So I decided to do something better: I observed that Bell Labs gave *every one* of its 20 or 30 thousand employees a brand new, *bound* company phone book every six months, with white pages (alphabetical) and yellow pages (by department) . . . the whole nine yards. And it wasn't just for expensive techies – it was for everyone. . . . So why not do the same with UNIX man-

uals, I asked myself. My management gave me the expected answer: we had no money to thusly "subsidize" all users of UNIX who were not in our department.

I appealed my case, and finally played the BTL phone book trump card, and, to my surprise, I won. I was authorized henceforth to issue a brand new, bound manual with each major release.

Using the BTL phone book as a model, I realized that I could not use the "perfect" binding – the manual would never stay open to any one page – so I decided to use a comb binding; but I liked the 6" x 9" size of the BTL phone book and decided to adopt it for the UNIX System III manual.

I modified the -man troff macros by adding the -rs1 option ("s" for "small") and re-typeset the entire manual. To my amazement, only a dozen or so pages "broke" and required fixes (and in most cases, it was just careless initial coding that simply needed cleaning up).

The new format was a *big* success, and, although I can't prove it, I believe it was the first widely distributed computer manual in that size. Then a while later, IBM came out with the PC and used the 6" x 9" format, and the rest, as they say . . . If my belief is true, then most paper computer manuals today owe their size to the Bell Labs phone book. It's not as good a story as the one that derives the size of the space shuttle boosters to the size of the hind quarters of a Roman war horse, but it's my very own.

I love these stories, among other things because they frequently demonstrate just how clueless large corporate entities are. Here's a final Ted Dolotta anecdote for this article.

Speaking of loose-leaf binders, you may be aware that before its breakup in 1983, the entire Bell System was standardized on 8.5" x 11" FOUR-ring binders – two at the top, two at the bottom. But because the rest of the world used 3-ring binders, to help communications with the outside world Bell System punched paper had 7 holes, so it would fit into either 4-ring or 3-ring binders. This worked OK until a newly minted manager on a rotational assignment at Western Electric (which then purchased or manufactured everything for the Bell System, from paper clips to Central Offices) noticed that all stockrooms had 7-hole paper but 4-ring binders. He fixed the situation by decreeing that henceforth all binders would be 7-ring. And it was done. And when Western Electric purchased things, it purchased them in multi-railroad-car lots.

Thanks, Ted.

Call for Nominations for Two Awards

by the USENIX Awards Committee

<awards@usenix.org>

The USENIX Lifetime Achievement Award

The USENIX Lifetime Achievement Award is to recognize and celebrate singular contributions to the UNIX community in both intellectual achievement and service that are not recognized in any other forum. The award itself is in the form of an original glass sculpture called "the Flame," and in the case of a team based at a single place, a plaque for the team office. The award is presented on June 21 at the Annual USENIX Technical Conference, in San Diego, CA.

Past recipients of the USENIX Lifetime Achievement Award are the Computer Science Research Group at the University of California at Berkeley (and a cast of thousands) for the BSD line; Van Jacobson, and Mike Lesk for their contributions to networking technology; Tom Truscott, Steve Bellovin, and Jim Ellis for their work in creating USENET; the Software Tools Users Group for popularizing a new vision of operating system software, offering a bridge to portability and power for those limited by proprietary operating systems; Brian Kernighan for the books, tools, and insights into the use of language as a bridge between people and machines; Tim Berners-Lee for spinning the Web that has helped to transform the Internet into a fundamental part of everyday life and for his continued evangelism on its behalf; and the X Window System Community at Large.

The Software Tools Users Group Award

The Software Tools Award recognizes significant contributions to the general community that reflect the spirit and character of those who came together to form the Software Tools Users Group (STUG). This is a cash award.

STUG and the Software Tools effort was characterized by two important tenets. The first was an extraordinary focus on building portable, reusable libraries of code shared among multiple applications on wildly disparate systems. The other tenet, shared with the UNIX community, is "renegade empowerment."

The Software Tools Users Group gave users the power to improve their environment when their platform provider proved inadequate, even when local management sided with the platform provider. Therefore, nominees for the Software Tools Award should exhibit one or both of these traits in a conspicuous manner: a contribution to the reusable code-base available to all, or provsion of a significant enabling technology directly to users in a widely available form.

Past recipients of this award are Michael Tiemann for the production of G++, the GNU C++ Compiler; Larry Wall in recognition of his major contributions to software portability and reuse of code embodied in the public domain Config program and the Perl language; John Ousterhout for Tcl/Tk, the software tools for which he is best known; and Udi Manber for turning algorithms into tools for searching and resource discovery.

How to Nominate

If you believe someone qualifies for either of these awards, we (the nominating committee) welcome your input. Please send us your nomination by April 15, 2000, and include your name, details of the achievement, and your reasons for making the nomination. Send email to *<awards@usenix.org>*.

The members of the awards committee are Andrew Hume, chair; Jon "maddog" Hall, Mike O'Dell, Dennis Ritchie, Margo Seltzer, and Ellie Young.

Board Meeting Summary

by Gale Berkowitz
Deputy Executive Director
and Ellie Young
Executive Director

The following is a summary of some of the actions taken by the USENIX Board of Directors from November 1999 through February 2000.

SAGE Certification

A proposal from the SAGE Executive Committee for $135,000 to fund the first phase of the SAGE Certification Program was approved. See "Certification 2000" on page 75 of this issue for more information on this.

Public Relations

It was agreed to allocate $60,000 for public relations for USENIX and SAGE for one year.

USENIX 25th Anniversary

A proposal to allocate up to $107,000 to commemorate and celebrate the 25th Anniversary of USENIX was approved. A compendium of the best papers from the USENIX proceedings and a special giveaway will be produced and given to members. At the USENIX Annual Technical Conference, the following activities will be added: a display of USENIX and computing-related memorabilia, accompanied by a trivia contest and game show, and enhanced reception(s).

Tutorial Instructors Remuneration

A proposal to offer a $10 per attendee bonus to instructors when attendance in a class exceeds 100 was approved.

Conferences

Dan Geer was charged with trying to find a champion/program chair for an embedded systems conference.

A proposal from Dejan Milojicic for USENIX to sponsor, along with IEEE TCOS and ACM SIGOPS, an annual workshop on experiences in building and using industrial systems software and application environments was approved. The first workshop will be co-located at OSDI, and the next at SOSP.

It was agreed that USENIX will be a co-sponsor for the WCSMA and Mobicom conferences.

Electronic Frontier Foundation

The $100,000 contribution to EFF approved by the Board at its previous meeting will be used to support legal efforts in the DVD DeCSS cases (which put at risk free expression and fair use of information) and the Bernstein encryption software case.

Next Meeting

The next meeting will be held on Tuesday, June 20, at the USENIX Annual Technical Conference in San Diego, California.

SAGE news

From the SAGE President

by Barbara Dijker

Barbara Dijker is currently SAGE president. She's been sysadminning for about 12 years and runs a couple of ISPs.

<barb@usenix.org>

In the past few months several projects and plans have started to come together. As a result, SAGE is coming of age – leaving behind its awkward adolescence and finding its own road.

In SAGE'S early years, there were many ideas and little focus. Over time, important projects were knocked off – those with the most broad support and someone dedicated enough to see them to completion. They include the SAGE section of this publication, the jobs-descriptions booklet, LISA-NT, the policy booklet, code of ethics, etc. Then there came time and need for reflection, focus, and strategy.

Within the membership there is probably still not a unanimous view as to whether system administration is a "profession," a "trade," or just a collection of programmers more interested in solving problems than cranking out code. However, despite not being formally adopted until 1997, the published stated purpose of SAGE has always been to "advance the status of computer system administration as a profession." As a result, SAGE has built a membership that is aligned with that purpose, to further the profession of system administration. Growing interest in efforts such as certification is an example of this.

Whether it is just time or a self-fulfilling prophecy, the end result is the same. SAGE is now in a position to really make headway in accomplishing the original stated goal. Aside from certification, there are other irons in the fire along these lines.

Education of system administrators will be getting a boost this year. We have the occupational-analysis research to support educational work. Thanks to Steve Johnson and the USENIX board, system administration is on the program for the CRA (*<http://www.cra.org>*) conference this July. The CRA conference is a unique opportunity to address academics about incorporating system administration into computer-science curricula. David Parter is coordinating that presentation.

Both education and certification are large projects. Add to them our longstanding jobs-descriptions document and the work Geoff Halprin has been doing on taxonomy. There is potential for these efforts to diverge as they progress. To prevent that and to gain momentum, there will be a summit before summer on those four areas of work. The summit will help lay some groundwork and foster communication among these projects.

Recognition is a key factor in "advancing the profession" and making certification viable. As part of a general mode shift, SAGE is engaging in a concerted public-relations and marketing effort. The goal of this effort is to raise the awareness of SAGE as an organization and system administration as a profession. This will support all the SAGE programs and endeavors. As part of this, SAGE acquired the *sage.org* domain. The SAGE Web site and those of other international SAGE groups can now be reached at *<http://www.sage.org>*.

The members of our profession are our best advocates. So one of our best means of raising awareness is expanding our reach of members. Thanks to many dedicated volunteers, more local groups and international groups are in the works.

This year an effort is being undertaken to finally merge the U.S. and Australia codes of ethics into one common code acceptable to all. If you don't have a local group in your area, now is the time to start one.

Finally, we have some new energy on the executive committee. System administrators in particular tend to be overworked and overcommitted. As a volunteer commitment, unfortunately the SAGE executive committee may be one of the first to go when it is time to shed the load. We welcome Bruce Alan Wynn to the exec to finish out this term.

In addition to all of this, the usual work continues: conferences, surveys, publications, etc. If your job doesn't have you working 80 hours a week already, check the SAGE Web site for projects needing volunteers. As you would expect, everything SAGE does is the result of dedicated volunteers and hardworking staff. Where we are now simply wouldn't have been possible without the hard work of those who have gone before, paving the way.

Certification 2000

by Barbara Dijker

SAGE President

<barb@usenix.org>

After years and years of debate, SAGE last year took the first step in tackling this issue. The system administrator occupational analysis was completed in December 1999. An executive summary of that analysis is published here and on the SAGE Web site at *<http://www.sage.org/cert/>*.

For SAGE, the purpose of the analysis was twofold:

- to conduct research on the scope of our field that can be used for education, certification, and other endeavors

- to determine if certification is viable and appropriate for our field

The long-awaited answer to the burning question is "yes." We can do certification and the need exists. In January the SAGE Executive Committee made the decision to continue forward with system-administrator certification.

In this article we'll first cover a strawman of the overall plan and then specifics of the short-term (this year) goals.

An overall certification program will need to have a core track with probably three levels of difficulties or progressions. Let's call each of these a "module," for lack of a better word. So there would be three core modules: beginner, intermediate, and advanced, for example. There may also be specialty modules: security, networks, databases, etc. The specialties may or may not have modules at different levels of difficulties.

Each module would have a set of required elements to achieve certification for that module. Each core module will likely have three elements:

- multiple-choice test
- performance-based evaluation
- one to be determined

The reason for multiple-choice test elements is their cost-effectiveness. The other elements will cover requirements not well suited to that type of testing. The third element could be another type of test, or it could be something like training credits, achieving N specialty modules, job experience, etc. As any set of modules progresses in difficulty, there would be a greater emphasis on analytical skills.

This overall plan is quite ambitious. However, the basic idea is required for the context of the first phase. The first phase will consist of:

- Development of a business plan that includes implementation, delivery, how to raise recognition, and a long-term sustainable financial model (it has to pay for itself).
- Establishment of a policy, procedures, and planning body.
- Development of an initial program policy, including defining training-vendor participation.
- Establishment of a test-development body.
- Development of at least the first element of the beginner module.

A fact of life is that multiple-choice tests are fast, easy, and relatively inexpensive to both develop and deliver. The first one can and will be done in parallel with development of the overall program structure and further definition and development of other module test elements.

So by the end of this year SAGE will have a basic program laid out and the first element, if not the whole module, implemented. The fun has just begun.

Executive Summary of the System Administrator Occupational Analysis

Purpose of Project

The System Administrators Guild (SAGE) works to promote the interests of system administrators, particularly with regard to their professional development. One strategy that SAGE is interested in pursuing is the development of a certification program. SAGE is also conducting activities in support of the education and training of system administrators. The Human Resources Research Organization (HumRRO) worked with SAGE throughout 1999 to conduct an occupational/job analysis that could be used to support both of these areas of interest. This is a summary of their report.

Overview of Approach

The occupational-analysis process used available materials, interviews, workshops, and a survey of system administrators to describe the core requirements of the occupation. First, lists of the tasks required for the job and the knowledge areas, skills, and abilities (KSAs) required to perform those tasks were developed. The lists were reviewed and revised by system administrators on existing SAGE committees and those who volunteered to participate in a series of workshops. The lists were incorporated into a Web-based survey that was administered to system administrators. Survey respondents indicated which tasks and KSAs were applicable to their jobs and, for those that were relevant, rated their relative importance to overall job performance. Respondents were also asked to answer several questions related to their background (e.g., job tenure) and to issues related to certification and professional-development activities. Analysis of the survey data was used to identify the most important tasks and KSAs across the system-administrator occupation and at various skill levels.

Three advisory groups assisted in the planning and implementation of various stages of the project. The SAGE Certification Subcommittee, with the assistance of SAGE staff, was HumRRO's primary source of guidance and input for the project. This group included four subject-matter experts. Additional support was provided by the SAGE Executive Committee and the SAGE Certification Advisory Committee, a group of approximately 48 professionals who had offered to assist in the occupational-analysis project.

Results

The survey was available on the USENIX Web site from October 17 to November 17, 1999. When the survey-administration window was closed, a total of 1,217 surveys had been received.

Although 1,217 survey responses were received, not all were suitable for inclusion in the analyses. For example, system administration was not the primary activity for 177 (14.6%) respondents. There were 30 (2.5%) respondents who had less than one year of experience as a system administrator. Because these 198 (16.3%) respondents had too little knowledge of system administration to make informed ratings of the importance of tasks and KSAs, they were excluded from all analyses. Deletion of these responses resulted in a final sample of 1,018 respondents.

Currently, there is no relatively comprehensive source of information about system administrators (e.g., who they are, how many there are). Therefore, the survey was not sent to specific potential respondents, but, rather, it was announced in locations system administrators would likely visit. Because of this, it is not possible to calculate a response rate (defined as the number of surveys returned divided by the number of surveys distributed). Nor is it possible to evaluate the representativeness of the obtained sample. That is, there is no way to tell whether, for example, the survey sample over- or underrepresents system administrators working with very large numbers of computers or particular operating systems.

Table 1. Sample Size and Years of Experience by System Administrator Level

		Years of Experience in System Administration			
Status	*Total*	*1–2*	*3–5*	*6–10*	*10 or more*
Beginner	127	97	24	5	1
Intermediate	471	0	250	153	68
Senior	420	5	75	161	179
TOTAL	1018	102	349	319	248

Table 1 shows the final sample sizes for each of the newly defined levels of system administrators, along with the years of experience reported by respondents at each level.

Table 2. Frequencies (Expressed as Percentages) of Level by Primary Setting

Primary Setting			*Level*	
	Total	*Beginner*	*Intermediate*	*Senior*
Commercial	41.4	46.5	41.2	40.0
Academic/Research	23.4	28.3	24.8	20.2
Consultant	10.4	7.1	9.8	12.1
Government	8.1	8.7	8.3	7.6
Financial	7.3	0.8	6.6	10.0
ISP	6.8	5.5	6.2	7.9
Military	2.8	3.1	3.2	2.1
TOTAL	100.0	100.0	100.0	100.0

Table 2 indicates that the largest proportion of the total sample (41%) work in a commercial setting, following by 23% who work in an academic or research setting.

Table 3 shows the major job responsibilities reported by the survey respondents. There were some interesting differences here among the three experience levels. For example, responsibilities related to security, development/programming, and Web hosting are more often assumed by senior administrators than by either intermediate or junior administrators.

Additional information about the 1,018 respondents indicates that almost 79% work at a high-complexity site. A high-complexity site is defined as one that supports 100 or more computers, 500 or more users, and/or 5 or more operating systems. Again, the more experienced administrators tend to work at the more complex sites.

Respondents marked which of several operating systems they use. Most respondents marked multiple systems. A large majority (76%) of the respondents report using Solaris and a smaller proportion (55%) use Linux. Slightly more than half (53%) of the respondents report using Windows NT. Almost 75% of the survey respondents work in the U.S.; the second-best-represented country was Australia, with 8.5%. The sample was 87.2% male and 88.2% white.

TABLE 3. FREQUENCIES (EXPRESSED AS PERCENTAGES) OF LEVEL BY MAJOR JOB RESPONSIBILITIES

Responsibilities			*Level*	
	Total	*Beginner*	*Intermediate*	*Senior*
General Sys Admin	96.5	96.1	97.7	95.2
Network Admin/Mgt	54.2	50.4	49.7	60.5
Security	44.8	35.4	40.1	52.9
Help Desk	30.1	34.7	29.9	28.8
Support Engineer	29.1	22.8	29.1	31.0
Development/Programming	24.4	18.9	21.0	29.8
Web Hosting	24.1	18.1	22.9	27.1
Database Admin/Mgt	21.8	19.7	19.5	25.0
Facilities Mgt	13.6	10.2	11.9	16.4
Sales	1.3	1.6	.6	1.9
Architecture	0.8	0.0	0.0	1.9
Project Mgt	0.4	0.0	0.2	0.7
Webmaster	0.2	0.8	0.0	0.2
Other	3.1	1.6	1.9	5.0

Note: Each value represents the percentage of respondents who indicated that they perform the responsibility listed in that row. Most respondents marked more than one responsibility.

The relative importance of tasks and KSAs was analyzed and ranked. The resulting ranking can be used as a blueprint for education and testing. For example, only those tasks and KSAs that ranked more important than others would be the focus of such efforts. The important KSAs are suitable for a blueprint of testing methods such as multiple choice. The important tasks are suitable for a blueprint of performance-based testing methodologies.

Needs Analysis Findings

Turning to the survey questions related to certification, respondents were given a list of seven different ways that a system-administrator certification program might affect them personally. The results were generally positive. Only 20% of the sample indicated that certification would either have no benefits or would have a negative effect. In contrast, 72% of the respondents believed that certification would help document their professional capabilities, and 56% believed that it could enhance their job prospects. As you would expect, senior system administrators perceive considerably fewer personal benefits to certification compared to their less experienced counterparts.

Respondents were also asked how a system-administrator certification program might affect the profession. Roughly 22% indicated that certification would have no benefits or could harm the profession. On the other hand, 63% of the total sample indicated that certification would likely increase professionalism and performance standards in the profession. There was a fairly consistent pattern in response to this item for the beginner system administrators to view certification more favorably than more experienced system administrators.

Although the majority of the respondents appeared to see benefits to professional certification, 20–22% is a significant minority of individuals who see no benefits and in some cases negative consequences. In open-ended comments invited at the end of the survey, a relatively large number were quite critical of plans for the development of a certification program.

Despite a fairly strong undercurrent of objections, a clear majority (74%) of the respondents indicated that they would probably or definitely participate in a SAGE certification program. Again, respondents in the beginner level showed the highest propensity to participate (83% probably or definitely participating), and senior-level respondents showed the lowest propensity to participate (66%).

Summary

Analysis of the survey data shows that the system-administration profession is amenable to certification testing. There appears to be a set of tasks and KSAs that are applicable to most system administrators. This assumes, however, that a test can be designed that does not require the examinees to know a specific operating system. The survey data also show that most system administrators support the idea of certification, although a vocal minority do not share this view. The ultimate success of the program would, of course, be determined by several factors such as the quality of the tests, the popularity of the program, its cost, and how well the program is administered.

The 1999 SAGE System Administrator occupational-analysis study has provided a considerable amount of detailed information that will be useful for a variety of purposes. The information can be used for certification, but can also be used as a foundation for other professional development tools and activities. Moreover, additional analyses can be performed to address other specific questions related to the profession.

Because system administration tends to see swift changes in technology, SAGE has been advised it will still be necessary to plan for updating the job analysis on a regular, and relatively frequent, basis. Once every year to two years, an updated automated survey could be administered. With the groundwork done in 1999, future job analysis will be considerably less burdensome. To further reduce the burden, it would be possible to reduce the amount of information collected on each survey, so that ratings on all the tasks and KSAs are not collected every time.

The certification subcommittee cannot complete this work in isolation from its members. Information on this project will continue to be disseminated as it becomes available. Members are encouraged to comment on this effort at <*sage-cert@usenix.org*> or the <*sage-members*> mailing list.

SAGE MEMBERSHIP
<office@usenix.org>

SAGE ONLINE SERVICES
Email server: <majordomo@usenix.org>
Web: <http://www.usenix.org/sage/>

SAGE SUPPORTING MEMBERS
Collective Technologies
Deer Run Associates
Electric Lightwave, Inc.
ESM Services, Inc.
GNAC, Inc.
Macmillan Computer Publishing, USA
Mentor Graphics Corp.
Microsoft Research
MindSource Software Engineers
Motorola Australia Software Centre
New Riders Press
O'Reilly & Associates
Remedy Corporation
Ripe NCC
SysAdmin Magazine
UNIX Guru Universe (UGU)

Announcement and Call for Participation

4th Annual Linux Showcase & Conference, Atlanta

Sponsored by USENIX, the Advanced Computing Systems Association, and the Atlanta Linux Showcase, in cooperation with Linux International

October 10-14, 2000, Cobb Galleria, Atlanta, Georgia

http://www.linuxshowcase.org

Important Dates

Extreme Linux abstracts due: *April 17, 2000*
Hack Linux/Use Linux abstracts due: *May 1, 2000*
Extreme Linux author notification: *June 16, 2000*
Hack Linux/Use Linux author notification: *June 30, 2000*
Registration material available: *July 2000*
Extreme Linux editorial revisions due: *July 23, 2000*
Final papers due (for all tracks): *August 24, 2000*

Program Committee

Hack Linux: Technical Development Track
Chair: Theodore Ts'o, *VA Linux Systems*
Bryan C. Andregg, *Red Hat*
Peter Beckman, *Los Alamos National Labs*
Jes Sorensen, *Linux Care*
Stephen Tweedie, *Red Hat*
Victor Yodaiken, *New Mexico Institute of Mining and Technology*
Leonard Zubkoff, *VA Linux Systems*

Use Linux: Tools and Applications Track
Chair: Greg Hankins, *Atlanta Linux Showcase*
Eric Ayers, *Atlanta Linux Enthusiasts*
Donnie Barnes, *Red Hat Software*
Jeff Carr, *LinuxPPC*
Clem Cole, *Compaq*
Danny Cox, *Atlanta Linux Showcase*
Hunter Eidson, *Atlanta Linux Showcase*
Vernard Martin, *Atlanta Linux Showcase*

Extreme Linux: Clusters and High Performance Computing Workshop
Co-Chairs: Pete Beckman, *Los Alamos National Laboratory*, and David Greenberg, *Center for Computing Sciences*
David Bader, *University of New Mexico*
Don Becker, *Scyld Computing Corporation and USRA-CESDIS*
Peter J. Braam, *Stelias Computing Inc & Carnegie Mellon University*
Remy Evard, *Argonne National Laboratories*
Jon "maddog" Hall, *VA Linux*
David Halstead, *Ames Laboratory*
Yutaka Ishikawa, *RWCP*
Walter Ligon, *Clemson University*
Greg Lindahl, *HPTi*
Bill Nitzberg, *NASA*
Bill Saphir, *LBL/NERSC*

Overview

The highlight of this year's Linux community calendar is undoubtably the Annual Linux Showcase and Conference. Now in its fourth year, ALS 2000 is specifically designed for the Linux enthusiast, with an emphasis on high-caliber technical knowledge. This conference is developed by a volunteer community of computing professionals and by USENIX, a not-for-profit technical association respected for its tradition of in-depth technical conferences.

ALS 2000 promises to be the biggest event in ALS history, expanding its technical program to include more tutorials, refereed papers, invited talks, Birds-of-a-Feather sessions, hothouses, and opportunities for informal discussions with Linux experts, professionals, and vendors. The conference includes a three-day vendor exhibition in which more than 80 companies will showcase their latest products and services.

Technical Sessions, Oct. 12-14

Three days of technical sessions feature three parallel tracks of refereed papers and invited talks. Refereed papers are published in Proceedings which are provided free to Technical Sessions attendees in print and a CD-ROM. Papers that analyze problem areas and draw important conclusions from practical experience are especially welcome.

Hack Linux: Technical Development Track

For this refereed technical track, we seek papers on topics relating to Linux and Open Source development, including:

- Kernel performance improvements and enhancements
- Linux networking
- Linux clustering and high availability
- Ports to new hardware architectures
- Low-level support infrastructures for X11, KDE, and GNOME
- Embedded Linux

Selection will be based on the quality of the written submission and whether the work advances the state of the art of Linux implementation. Please see also the detailed author guidelines of the Web site, including sample extended abstracts and final papers.

Use Linux: Tools and Applications Track

Paper submissions should focus on the following Linux tools and applications areas:

- Development
- Networking and running network services
- System administration
- Usability and desktop interfaces
- Security
- Integration and interoperability with non-Linux systems
- Performance tuning

Extreme Linux: Clusters and High Performance Computing Workshop

This track is devoted to clusters using Linux which are intended to scale to hundreds or thousands of processors. Comparisons with other systems are of interest, but papers should directly address issues of Linux and scale. For a complete Call for Papers for this track, see *http://www.extremelinux.org*.

How to Submit

The submission deadline for the Hack Linux and Use Linux Tracks is May 1, 2000. The submission deadline for maximum of ten page extended abstracts for the Extreme Linux Track is April 17, 2000. All submissions will be electronic via a web submission form available on the conference Web site.

Submissions may be formatted in any way that you find convenient. However, accepted papers will need to be submitted in the format prescribed by USENIX. If you would like to avoid future formatting changes, you may consult predefined templates on the conference Web site which format according to the USENIX guidelines.

Further details for specific tracks can be found below. Specific questions about submissions may be sent to the program chair via email to:

Hack Linux Track:
hacklinuxchair@usenix.org
Use Linux Track:
uselinuxchair@usenix.org
Extreme Linux Workshop:
extremelinuxchairs@usenix.org

Guide to Submissions for Hack Linux and Use Linux Tracks

Submissions should be 2-5 page summaries in ASCII, Postscript, or PDF format. You may submit a full paper, however we expect that most submissions will be 2-5 page summaries of your work to date. In no event should you submit a description in excess of 14 pages. However, please provide enough detail to let us know what you are doing; one paragraph summaries are unlikely to be accepted.

A good submission will clearly demonstrate that the authors:

- are attacking a significant problem,
- are actively working on a solution,
- have made enough progress to have useful information to report.

In particular, the technical tracks are not for people that are thinking about doing some project, but have not yet started it. Such talks are better presented in the Work-In-Progress (WIP) session.

Authors will be notified by June 30, 2000. All accepted submissions will be expected to produce a written report for the proceedings by the August 24, 2000 deadline. These reports need not be polished papers, but they should describe work that has been completed as of the time of their submission. The purpose of your paper is to let readers and attendees know what you are doing. Your talk at the conference may describe not only what is in your paper but also the work completed between the time that the paper is submitted and the conference is held. Members of the program committee will help shepherd authors through the writing process prior to final acceptance for publication in the proceedings.

Each accepted paper must be presented at the conference by at least one author. Authors of an accepted paper must provide a final paper for publication in the conference proceedings and electronic files for the CD-ROM and Web site. Final papers are limited to 12 pages, including diagrams, figures and appendices.

For the Hack Linux Track, please see the author guidelines on the Web site, including sample extended abstracts and final papers: *http://www.linuxshowcase.org/cfp/guidelines.html.*

Guide to Submission of Extended Abstracts for Extreme Linux Track

All submissions for the Extreme Linux track of ALS 2000 will be electronic, in PostScript or PDF. Authors will be notified of receipt of submission via e-mail. If you do not receive notification, contact: *extremelinuxchairs@usenix.org.*

Extended abstracts, up to 10 pages in length, are preferred, but full papers can also be submitted. Papers should be 8 to 12 single-spaced 8.5 x 11 inch pages (about 4000-6000 words), not counting figures and references. Papers longer than 14 pages will not be reviewed.

It is imperative that you follow the instructions for submitting a quality paper. A good paper will clearly demonstrate that the authors:

- are attacking a significant problem,
- are familiar with the literature,
- have devised an original or clever solution,
- if appropriate, have implemented the solution and characterized its performance using reasonable experimental techniques, and
- have drawn appropriate conclusions from their work.

In order to facilitate the submission of current work the Extreme Linux Track will use the "shepherded paper" model. The program committee will notify authors of the acceptance or rejection of their paper by June 16, 2000. A program committee member will help the authors to produce a second version of the paper which may include updated results by July 23, 2000. Final, camera ready, versions must be submitted by August 24, 2000 in order to be including in the printed proceedings.

Note: the ALS Extreme Linux Track, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers submitted to this conference that are under review elsewhere will not be reviewed. Papers accompanied by non-disclosure agreement forms can not be accepted, and will not be reviewed. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Cash Prizes

The USENIX Association will award cash prizes at the conference for the best paper and for the best paper by a student.

Financial Assistance

Some financial assistance is available for speakers' travel and accommodation in cases of need. All speakers will receive free admission to the conference and an invitation to the welcome dinner on Wednesday evening.

Tutorials, Oct. 10-11

Gain mastery of complex techniques and technologies, and you'll get immediate payoff within your organization. There will be full and half-day tutorials in all areas and levels of expertise of Linux. If you are interested in presenting a tutorial at ALS, contact the tutorial coordinators: Daniel V. Klein/Paul Manno, Email: *alstutorials@linuxshowcase.org*

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are very informal gatherings organized by attendees interested in a particular topic. BoFs are held in the evenings. BoFs may be scheduled in advance via email to *alsbofs@usenix.org.* or at the conference.

Works-in-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The ALS audience provides valuable discussion and feedback. Presentations of student work are particularly welcome. To schedule your short report, send email to *alswips@usenix.org.*

Exhibition, Oct. 12-14

In the exhibition, the emphasis is on serious questions and feedback. Vendors will demonstrate the features and technical innovations that distinguish their products. For more information, contact: Dana Geffner, 1.831.457.8649, *dana@usenix.org.*

Program and Registration Info.

Program and registration information will be available in July 2000 at the conference Web site: *http://www.linuxshowcase.org.* If you would like to be kept up to date on ALS information, sign up for our email list by sending the message "subscribe als-announce" to *majordomo@linuxshow.org* or contact the USENIX Conference Office (email: *conference@usenix.org;* telephone: 1.949.588.8649).

Tutorial Program

2000 USENIX Annual Technical Conference

June 18-23, 2000, Marriott Hotel & Marina, San Diego, California, USA
Register online: http://www.usenix.org/events/usenix2000

SUNDAY

S1	**UNIX Security Tools: Use and Comparison** Matt Bishop, *University of California, Davis*
S2	**Sendmail Configuration and Operation (Updated for Sendmail 8.10)** Eric Allman, *Sendmail, Inc.*
S3	**System and Network Performance Tuning** Marc Staveley, *Sun Microsystems, Inc.*
S4	**Advanced Topics in Perl Programming** NEW Tom Christiansen, *Consultant*
S5	**Windows NT Internals** Jamie Hanrahan, *Kernel Mode Systems*
S6	**Hacking Exposed: LIVE!** NEW George Kurtz and Eric Schultze, *Rampart Security Group*
S7	**Introduction to UNIX Administration** Peter Baer Galvin, *Corporate Technologies, Inc.*
S8	**Cryptographic Algorithms Revealed** NEW Greg Rose, *QUALCOMM Australia*

MONDAY

M1	**Intrusion Detection and Network Forensics** Marcus J. Ranum, *Network Flight Recorder, Inc.*
M2	**Advanced Solaris Systems Administration Topics** Peter Baer Galvin, *Corporate Technologies, Inc.*
M3	**Linux Systems Administration** Bryan C. Andregg, *Red Hat, Inc.*
M4	**Windows NT and UNIX Integration: Problems and Solutions** Phil Cox, *SystemExperts Corporation*; Gerald Carter, *Auburn University*
M5	**Security from the Inside Out: System Engineering for Security Systems** NEW Char Sample, *L-3 Network Security*; Ian Poynter, *Jerboa Inc.*
M6	**Topics in Systems Administration I** NEW Ned McClain, *XOR Network Engineering*; Evi Nemeth, University of Colorado
M7	**Administering Windows 2000: A Course for UNIX People** UPDATED Aeleen Frisch, *Exponential Consulting*
M8	**Advanced CGI Techniques Using Perl** NEW Tom Christiansen, *Consultant*
M9	**Modern Security Systems for Intranets, Extranets, and the Internet** Daniel E. Geer, Jr., *@Stake*; Jon Rochlis, *SystemExperts Corporation*
M10	**Secure Networking: An Introduction to VPN Architecture and Implementation** NEW Tina Bird, *Counterpane Internet Security*

TUESDAY

T1	**Designing Resilient Distributed Systems—High Availability** Evan Marcus, *VERITAS Software Corporation*
T2	**Solaris Internals: Architecture, Tips, and Tidbits** Richard McDougall and James Mauro, *Sun Microsystems, Inc.*
T3	**Inside the Linux Kernel** Stephen C. Tweedie, *Red Hat, Inc.*; Theodore Ts'o, *VA Linux Systems*
T4	**Configuring and Administering Samba Servers** NEW Gerald Carter, *Auburn University*
T5	**Computer Attacks: Trends and Countermeasures** Tina Darmohray, *Consultant*; Phil Cox, *SystemExperts Corporation*
T6	**Network Administration** NEW Bryan C. Andregg, *Red Hat, Inc.*
T7	**Practical Web Site Development and Maintenance with Perl: A Cookbook Approach** NEW Mark-Jason Dominus, *Consultant*
T8	**Managing and Being Managed** NEW Steve Johnson, *Transmeta*; Dusty White, *Consultant*
T9	**Network Security Profiles: A Collection (Hodgepodge) of Stuff Hackers Know About You** Brad Johnson, *SystemExperts Corporation*
T10	**Special Topics in Sendmail: Sendmail 8.10 and Sendmail Security** NEW Eric Allman and Gregory Neil Shapiro, *Sendmail, Inc.*

For tutorial descriptions, see: http://www.usenix.org/events/usenix2000

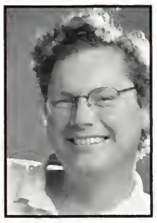
Technical Program

2000 USENIX Annual Technical Conference

June 18-23, 2000, Marriott Hotel & Marina, San Diego, California, USA
Register online: http://www.usenix.org/events/usenix2000

Refereed Papers | Invited Talks | FREENIX

Wednesday, June 21, 2000



Opening Remarks and Keynote

Keynote Address

Bill Joy, *Sun Microsystems Co-Founder and Vice President*

Bill Joy will be talking about his vision of the future of computing.

Instrumentation and Visualization

Session Chair: Christopher Small, *Bell Labs Research, Lucent Technologies*

Mapping and Visualizing the Internet
Hal Burch, *Carnegie Mellon University;* Bill Cheswick and Steve Branigan, *Bell Labs Research, Lucent Technologies*

Measuring and Characterizing System Behavior Using Kernel-Level Event Logging
Karim Yaghmour and Michel R. Dagenais, *Ecole Polytechnique de Montréal*

Pandora: A Flexible Network Monitoring Platform
Simon Patarin and Mesaac Makpangou, *INRIA Rocquencourt*

Computer System Security: Is There Really a Threat?

Avi Rubin, *AT&T Research*

I'm often asked, "If we're so vulnerable, how come I don't hear about incidents that often?" While I cannot answer that question, I can try to answer the question of whether or not there is a threat. In this talk, I will look at some historic and some more recent computer security incidents. How did the attacks occur? Why did they succeed? What were the consequences? Could it have been worse? We will look at security issues in existing systems and assess the level of danger. Finally, I'll discuss what the best defenses are, and the steps we can each take to secure our systems and data.

Storage Systems

Session Chair: Marshall Kirk McKusick, *Author & Consultant*

Swarm: A Log-Structured Storage System for Linux
Ian Murdock and John H. Hartman, *University of Arizona*

DMFS—A Data Migration File System for NetBSD
William Studenmund, *Veridian MRJ Technology Solutions*

A 3-Tier RAID Storage System with RAID1, RAID5, and Compressed RAID5 for Linux
K. Gopinath, *IISc;* Nitin Muppalaneni, *VERITAS Software;* N. Suresh Kumar, *Lucent Technologies;* Pankaj Risbood, *IISc*

File Systems

Session Chair: Liuba Shrira, *Brandeis University*

A Comparison of File System Workloads
Drew Roselli and Jacob R. Lorch, *University of California at Berkeley;* Thomas E. Anderson, *University of Washington*

FiST: A Language for Stackable File Systems
Erez Zadok and Jason Nieh, *Columbia University*

Logging Versus Soft Updates: Asynchronous Metadata Protection in File Systems
Margo I. Seltzer, *Harvard University;* Gregory R. Ganger, *Carnegie Mellon University;* M. Kirk McKusick, *Author & Consultant;* Keith A. Smith, *Harvard University;* Craig A. N. Soules, *Carnegie Mellon University;* Christopher A. Stein, *Harvard University*

Watching the Waist of IP

Steve Deering, *Cisco Systems*

The Internet protocol architecture has an hourglass shape: a wide variety of applications and end-to-end (upper-layer) protocols are supported by a single, "narrow" protocol called IP, which in turn rests upon a wide variety of network and datalink (lower-layer) protocols. The Internet's enormous flexibility in accommodating new transmission technologies and new applications, and its ability to serve as the convergence platform for data, telephony, TV, and other media, depend on this hourglass design. However, as the Internet has grown, the waist of the hourglass has spread. In this talk, I review the evolution of the IP layer of the Internet, discuss the consequences of the changes, and speculate on the future shape of IP.

Network Systemt Administration

Session Chair: Victor Yodaiken, *FSMLabs and New Mexico Institute of Technology*

Extending Internet Services Via LDAP
James Dutton, *Southern Illinois University at Carbondale*

MOSIX: How Linux Clusters Solve Real-World Problems
Steve McClure and Richard Wheeler, *EMC*

Webmin
Jamie Cameron, *Caldera Systems*

Register online: http://www.usenix.org/events/usenix2000

Technical Program

2000 USENIX Annual Technical Conference

June 18-23, 2000, Marriott Hotel & Marina, San Diego, California, USA
Register online: http://www.usenix.org/events/usenix2000

Refereed Papers

Old Dogs, New Tricks
Session Chair: Greg Minshall, *Siara Systems*

Lexical File Names in Plan 9, or, Getting Dot-Dot Right
Rob Pike, *Lucent Technologies—Bell Labs*

Gecko: Tracking a Very Large Billing System
Andrew Hume, *AT&T Labs—Research;* Scott Daniels, *EDS;* Angus MacLellan, *AT&T Labs*

Extended Data Formatting Using Sfio
David G. Korn, Glenn S. Fowler, and Kiem-

Invited Talks

Phong Vo, *AT&T Labs—Research*

FREENIX

Implementing 3D Workstation Graphics on PC UNIX Hardware
Daryll Strauss, *Precision Insight*

3D hardware for PCs has improved to the point that it is beginning to rival that of traditional 3D graphics workstations. Providing these capabilities on commodity hardware poses a number of difficult problems. For example, 3D hardware has a voracious appetite for data, and commodity hardware is typically not designed for secure multitasking. Precision Insight is working with a number of vendors to provide completely open-source solutions to these problems under X and Linux.

Thursday, June 22, 2000

Distribution and Scalability: Problems and Solutions
Session Chair: Ken Arnold, *Sun Microsystems*

Virtual Services: A New Abstraction for Server Consolidation
John Reumann, *University of Michigan;* Ashish Mehra, *IBM TJ Watson Research;* Kang Shin, *University of Michigan;* Dilip Kandlur, *IBM TJ Watson Research*

Location-Aware Scheduling with Minimal Infrastructure
John Heidemann and Dhaval Shah, *USC/ISI*

Distributed Computing: Moving from CGI to CORBA
James FitzGibbon and Tim Strike, *Targetnet.com Inc.*

The Microsoft Antitrust Case: A View from an Expert Witness
Edward Felten, *Princeton University*

Edward Felten recently served as an expert witness in the Microsoft antitrust case, and as a consultant to the Department of Justice. He will talk about his experiences in working on this high-profile case, and what he learned about the law, economics, computer science, and connections among them.

Sockets
Session Chair: David Greenman, *The FreeBSD Project*

Protocol Independence Using the Sockets API
Craig Metz, *University of Virginia*

Scalable Network I/O in Linux
Niels Provos, *University of Michigan;* Chuck Lever, *Netscape Communications Corp.*

"Thundering Herd" Issues in Linux accept(2)
Stephen Molloy and Peter Honeyman, *CITI, University of Michigan;* Chuck Lever, *Sun-Netscape Alliance*

Tools
Session Chair: Eran Gabber, *Lucent Technologies—Bell Labs*

Outwit: UNIX Tool-Based Programming Meets the Windows World
Diomidis Spinellis, *University of the Aegean*

Plumbing and Other Utilities
Rob Pike, *Lucent Technologies—Bell Labs*

Integrating a Command Shell into a Web Browser
Robert C. Miller and Brad A. Myers, *Carnegie Mellon University*

Challenges in Integrating the Mac OS and BSD Environments
Wilfredo Sanchez, *Apple Computer*

Apple's next-generation operating system, Mac OS X, is a drastic departure from previous versions of the Mac OS. Mac OS X's core operating system is a derivative of BSD UNIX, topped by a suite of application toolkits. The user-friendly GUI of the original Mac OS has been widely emulated in the personal computer industry. BSD's robust core, advanced networking, and scalability are highly valued in engineering and server applications. The combination offers a great deal of promise, but it has required many changes in the architecture of system components. Additionally, users use the systems in very different ways and expect different sorts of behavior.

Network Publishing
Session Chair: Chris Demetriou, *AT&T Labs*

Making Web Publishing Irreversible
David S. H. Rosenthal and Victoria A. Reich, *Stanford Libraries*

Globe and the Globe Distribution Network
Arno Bakker, Egon Amade, Gerco Ballintijn, Ihor Kuz, Patrick Verkaik, Ivo van der Wijk, Maarten van Steen, and Andrew Tanenbaum, *VU Amsterdam*

Open Information Pools
Johan Pouwelse, *Delft University of Technology*

Technical Program

2000 USENIX Annual Technical Conference

June 18-23, 2000, Marriott Hotel & Marina, San Diego, California, USA
Register online: http://www.usenix.org/events/usenix2000

Refereed Papers

Kernel Structures

Session Chair: Keith A. Smith, *Harvard University*

Operating System Support for Multi-User, Remote, Graphical Interaction
Alexander Ya-li Wong and Margo Seltzer, *Harvard University*

Java Operating Systems: Design and Implementation
Godmar Back, Patrick Tullmann, Wilson C. Hsieh, and Jay Lepreau, *University of Utah*

Signaled Receiver Processing
José Brustoloni, Eran Gabber, Abraham Silberschatz, and Amit Singh, *Lucent Technologies—Bell Labs*

Invited Talks

The Convergence of Networking and Storage: Will It Be SAN or NAS?

Rod Van Meter, *Network Alchemy*

What we think of as storage generally follows one of two models—either named files or undifferentiated, numbered blocks. Both models can be presented on a network. The former is often called network-attached storage (NAS); the latter, storage-area networks (SAN). This talk will explore the differences and similarities between the two and will examine where both are likely to go in the near future. Emphasis will be on scalability, naming, security, and network media.

FREENIX

X11 and User Interfaces

Session Chair: Miguel de Icaza, *Helix Code, Inc.*

The GNOME Canvas: A Generic Engine for Structured Graphics
Federico Mena-Quintero, *RHAD Labs*; Raph Levien, *Code Art Studio*

Efficiently Scheduling X Clients
Keith Packard, *SuSE, Inc.*

Developing Drivers and Extensions for XFree86-4.x
Dirk Hohndel, *SuSE Linux AG*

Works in Progress Reports (WIPs)

Session Chair: Aaron Brown, *University of California at Berkeley*

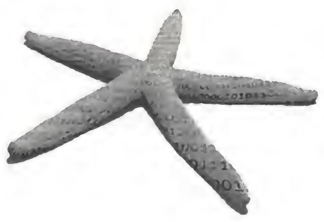
Pithy and fun, Works in Progress Reports introduce interesting new or ongoing work, and the USENIX audience provides valuable discussion and feedback.

Slots are limited. If you have interesting work you'd like to share, or a hot idea that's not yet ready for publication, send a paragraph or two of description to Aaron Brown at *usenix2000-wips@usenix.org*. Student work is particularly welcome.

Lessons Learned About Open Source

Jim Gettys, *Compaq*

The X Window System was developed open-source using the Internet from nearly its inception, but has taken a number of (partial) turns along the way. These were partly forced by commercial pressure, but primarily because the Internet was not able to support the kind and scale of development seen in free software today. Now we see large-scale open-source software engineering with hundreds of contributors to a given project. Amazingly, X is alive and moving forward again. What can we learn from these experiences? What traps can be avoided? What opportunities are offered by the new desktops and new window managers? Where is further work needed? How should we further exploit the Web? What is possible now that we have more developers for open source than sit behind the walls of any corporation on the planet?



Friday, June 23, 2000

Run-Time Tools and Tricks

Session Chair: Christopher Small, *Bell Labs Research, Lucent Technologies*

DITools: Application-Level Support for Dynamic Extension and Flexible Composition
Albert Serra, Nacho Navarro, and Toni Cortes, *Universitat Politècnica de Catalunya*

Portable Multithreading—The Signal Stack Trick for User-Space Thread Creation
Ralf S. Engelschall, *TUM*

Transparent Run-Time Defense Against Stack-Smashing Attacks
Arash Baratloo, Timothy Tsai, and Navjot Singh, *Bell Labs Research, Lucent Technologies*

An Introduction to Quantum Computation and Communication

Rob Pike, *Lucent Technologies—Bell Labs*

Quantum computation is more than just the use of very small things to compute. It exploits the fundamentally odd properties of quantum-mechanical interaction to achieve profound parallelism, zero-energy calculations, and other technological marvels. I will discuss how the quantum world makes these things possible, the design of quantum hardware and software, proposals for practical quantum devices, and the prospects for quantum computation and communication in our lifetimes.

Security

Session Chair: Niels Provos, *University of Michigan*

Implementing Internet Key Exchange, IKE
Angelos D. Keromytis, *University of Pennsylvania;* Niklas Hallqvist, *Applitron Datasystem AB*

Transparent Network Security Policy Enforcement
Angelos D. Keromytis, *University of Pennsylvania;* Jason Wright, *University of North Carolina at Greensboro*

Safety Checking of Kernel Extensions
Craig Metz, *University of Virginia*

Refereed Papers

Measurement and Stability

Session Chair: Fred Douglis, AT&T Labs—Research

Towards Availability Benchmarks: A Case Study of Software RAID Systems
Aaron Brown and David A. Patterson, *University of California at Berkeley*

Performing Replacement in Modem Pools
Yannis Smaragdakis, *Georgia Tech;* Paul Wilson, *University of Texas at Austin*

Auto-Diagnosis of Field Problems in an Appliance Operating System
Gaurav Banga, *Network Appliance*

Servers: Load Balancing and Scheduling

Session Chair: Yoonho Park, *IBM Research*

Dynamic Function Placement for Data-Intensive Cluster Computing
Khalil Amiri, David Petrou, and Greg Ganger, *ECE, Carnegie Mellon University;* Garth Gibson, *CS, Carnegie Mellon University*

Scalable Content-Aware Request Distribution in Cluster-Based Network Servers
Mohit Aron, Darren Sanders, Peter Druschel, and Willy Zwaenepoel, *Rice University*

Isolation with Flexibility: A Resource Management Framework for Central Servers
David G. Sullivan and Margo I. Seltzer, *Harvard University*

Invited Talks

Providing Future Web Services

Andy Poggio, *Sun Labs*

This presentation will begin by describing the day when desktop PCs will no longer dominate as networked devices. In this new era, network appliances will be the most common devices. It will discuss Web services for commerce, education, and entertainment: how they'll change, and what new Web services will proliferate. Finally, it will describe in detail the computer system architecture and network infrastructure that will be needed to provide these services, including the roles that InfiniBand, IPv6, and other new technologies will play.

The GNOME Project

Miguel de Icaza

The GNU Network Object Model Environment (GNOME) project aims at providing a framework for UNIX application development. Lack of infrastructure has made UNIX systems lag in some areas. GNOME provides a component model that encourages code reuse and tool replacement by making applications adhere to a set of GNOME-standardized CORBA interfaces. A name server and an object-launching facility are used to make GNOME tools integrate in the desktop. GNOME graphical applications are written using the GTK+ toolkit, and they use the GNOME foundation libraries to simplify programming and encourage a standardized graphical user environment. The GNOME printing subsystem provides programmers with a portable and powerful printing subsystem.

FREENIX

Cool Stuff

Session Chair: Clem Cole, *Compaq*

An Operating System in Java for the Lego Mindstorms RCX Microcontroller
Pekka Nikander, *Helsinki University of Technology*

LAP: A Little Language for OS Emulation
Donn Seeley, *Berkeley Software Design, Inc.*

Traffic Data Repository at the WIDE Project
Kenjiro Cho, *Sony Computer Science Laboratories, Inc.;* Koushirou Mitsuya, *Keio University;* Akira Kato, *The University of Tokyo*

Short Topics

Session Chair: Stephen C. Tweedie, *Red Hat, Inc.*

JEmacs—The Java/Scheme-Based Emacs
Per Bothner, *Consultant*

A New Rendering Model for X
Keith Packard, *SuSE, Inc.*

UBC: An Efficient Unified I/O and Memory Caching Subsystem for BSD
Chuck Silvers, *VERITAS Software*

Mbuf Issues in 4.4BSD IPv6 Support—Experiences from the KAME Project
Jun-ichiro Hagino, *Research Laboratory, IIJ*

Malloc() Performance in a Multithreaded Linux Environment
Chuck Lever and David Boreham, *Netscape Communications Corp.*

The AT&T OpenSource Software Collection
Glenn Fowler, David Korn, Stephen North, and Kiem-Phong Vo, *AT&T Labs—Research*

Closing Session

New Horizons for Music on the Internet
Thomas Dolby Robertson, *Beatnik, Inc.*

The dynamics of creating and experiencing Web content are continually evolving. The integration of music and interactive audio into the fabric of computer and Internet technologies have enhanced the overall Web experience, moving it from a silent environment to a multi-sensory one. Come see what Thomas Dolby Robertson and his company, Beatnik, Inc., have contributed to the world of the Internet using sound and audio technologies. Mr. Robertson will show that everyone, from composers and musicians to Web homesteaders and professional Web designers, can benefit from these evolving technologies. Case studies presented will also illustrate how the emergence of new applications is making the Web a stage for true musical interaction.

Register online: http://www.usenix.org/events/usenix2000

Announcement and Call for Papers **USENIX**

6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS '01)

http://www.usenix.org/events/coots01

January 29 - February 2, 2001 **San Antonio, Texas, USA**

Important Dates

Paper submissions due: *July 27, 2000*
Tutorial submissions due: *July 27, 2000*
Notification of acceptance: *September 11, 2000*
Camera-ready final papers due: *December 5, 2000*

Conference Organizers

Program Chairs
Rajendra Raj, *Morgan Stanley Dean Witter*
Yi-Min Wang, *Microsoft Research*

Program Committee
Mustaque Ahamed, *Georgia Tech.*
Ken Arnold, *Sun Microsystems*
Don Box, *DevelopMentor*
Murthy Devarakonda, *IBM T.J. Watson Research Center*
Rachid Guerraoui, *Swiss Federal Institute of Technology, Switzerland*
Jennifer Hamilton, *Microsoft Corporation*
Eric Jul, *University of Copenhagen, Denmark*
Doug Lea, *State University of New York at Oswego*
Keith Marzullo, *University of California, San Diego*
Ira Pohl, *University of California, Santa Cruz*
Douglas C. Schmidt, *Washington University*
Christopher Small, *Bell Laboratories, Lucent Technologies*
Robert Stroud, *University of Newcastle upon Tyne, UK*
Bjarne Stroustrup, *AT&T Labs*
Joe Sventek, *Agilent Laboratories, UK*
Steve Vinoski, *IONA Technologies, Inc.*
Werner Vogels, *Cornell University*
Shalini Yajnik, *Bell Laboratories, Lucent Technologies*
Deborra Zukowski, *IBM T.J. Watson Research Center*

Tutorial Program Chair
Douglas C. Schmidt, *Washington University*

Advanced Workshops Chair
Murthy Devarakonda, *IBM T.J. Watson Research Center*

Overview

As the first COOTS of the twenty-first century, COOTS '01 invites quality papers describing research or experience with object technology. Research papers should describe original work that offers significant contributions to the state of object technology. Experience papers should describe general insights gained from object technology practice. Submitted papers should make substantial contributions to the field and should be useful to researchers and practitioners alike.

This conference will last 5 days. Two days of tutorials will be followed by two days of technical sessions including refereed papers, guest lecturers, and a Work-in-Progress session(s). The last day of the conference will feature two Advanced Topics Workshops.

Topics

Relevant topics for COOTS '01 include, but are not limited to, the following:

- distributed systems and applications
- objects on the web
- component technologies
- middleware
- frameworks
- object databases and persistence
- systems testing, measurement and performance
- reliability
- security
- analysis and design methods
- design patterns
- language design and implementation
- programming techniques and application experiences
- development platforms, environments and tools

Questions about the relevance of a topic may be addressed to the Program Chairs at *coots01chairs@usenix.org*

Keynote Speaker and Guest Lecturers

In the tradition of USENIX conferences, COOTS '01 will feature prominent speakers who use their extraordinary insights, original thinking, creativity, and years of experience to help make a difference in the way we think about, design, develop, and deploy large software systems. In addition to the keynote speaker, this conference will feature guest lectures by well-known speakers on topics relevant to object technology practitioners during the sessions of the technical program.

see over →

Tutorials

On January 29 and 30, the COOTS conference will begin with tutorials. Tutorial topics may include distributed object systems (CORBA, COM+/Windows DNA, Java RMI/Jini, etc.); component technologies; web technologies (XML, web servers, etc.); framework design; and object-oriented languages.

If you are interested in proposing a tutorial, contact the USENIX tutorial coordinator: Dan Klein, Email: *dvk@usenix.org*, Phone: +1.412.422.0285.

Technical Sessions

On January 31 and February 1, the technical sessions will follow the tutorials. COOTS emphasizes research and advanced engineering aspects of object technology, focusing on experimental systems research. Conference Proceedings containing all refereed papers will be distributed to attendees and, following the conference, will be available online to USENIX members and for purchase. An award will be given for the best student paper at the conference.

Work-In-Progress Abstracts

This year, COOTS will include new session(s) on "work in progress" (WIP) to introduce new ideas to the community and solicit early feedback. We are particularly interested in the presentation of student work and bleeding edge usage of objects in industry. WIP abstracts will be lightly screened to facilitate focused discussions during these sessions. The submission process for WIP abstracts will begin in Sept. 2000. Full submission information will be available at the conference Web site.

Advanced Topics Workshops

The conference will conclude with two Advanced Topics Workshops, where smaller audiences can exchange in-depth technical information on a few position papers. The topics will be finalized and made available on the conference web site.

Attendance at the workshop is limited to the attendees of the main technical program and is based on acceptance of a position paper. As in past years, tutorial presenters, invited speakers, and authors of accepted papers in the technical program will also be invited to attend the workshop of their choice.

Potential workshop attendees are invited to submit a position paper in ASCII text of at most three (3) pages via electronic mail to the Workshops Chair no later than Dec 1, 2000. Acceptance notices to the authors will be issued by Dec 20, 2000. Position papers should briefly describe experiences, work in progress, and/or ongoing research and development in the topic area. A representative subset of authors of position papers may be invited to make informal presentations at the workshops.

If you have any questions regarding the topics, especially if you are concerned whether your focus is relevant to the chosen topics, do not hesitate to send electronic mail to the Workshops Chair at *coots01ATWchair@usenix.org*.

What to Submit

Full papers should be 10 to 15 pages (around 5,000-6,000 words). Papers that are too long or are late will be rejected. All submissions will be judged on originality, significance, relevance, correctness, and clarity. Each submission must include a cover letter stating the paper title, the contact author, email and regular addresses, and a phone number. Please see the detailed guidelines for submission at the conference Web site.

The COOTS conference, like most conferences and journals, requires that papers not be submitted simultaneously to any other conference or publication, that submissions not be previously published, and that accepted papers not be subsequently published elsewhere. Papers accompanied by nondisclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

How to Submit

Web-based electronic submission will be expected. Submissions should be in Postscript that is interpretable by Ghostscript or in PDF that is interpretable by Acroread, and should be printable on US Letter sized paper. A Web form for submissions will be available in January 2000 on the conference Web site at *http://www.usenix.org/events/coots01*. All submissions will be acknowledged.

Submitters for whom web submission is a hardship should contact the Program Chairs for alternative means of submission at *coots01chairs@usenix.org*.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available by October 2000 at the conference Web site. If you wish to receive the registration materials in print, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

Rev. 9/21/99

USENIX <u>**The Advanced Computing Systems Association**</u>

MEMBERSHIP INFORMATION

Indicate the category of membership which you prefer and send appropriate annual dues

☐	*Individual:*	*$ 95*
☐	*Full-time Student:* Please enclose a copy of your current student ID.	*$ 25*
☐	*Educational:*	*$ 200*
☐	*Corporate:*	*$ 400*
☐	*Supporting – USENIX –* ☐ *$ 1000* ☐ *$ 2500*	*SAGE –* ☐ *$ 1000*

A designated representative for each Educational, Corporate, and Supporting membership receives all USENIX conference and symposia proceedings published during their membership term plus all member services. Supporting members also receive one free full page ad in *;login:*, more member-rate discounts for technical sessions, a half-price rental of the mailing list, and a link to their URL from the USENIX Web site.

$50 of your annual membership dues is for a one-year subscription to *;login:*

SAGE THE SYSTEM ADMINISTRATORS GUILD

The System Administrators Guild

SAGE, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

☐ *Individual: $30* ☐ *Students: $15*

MEMBERSHIP APPLICATION: ☐ *New* ☐ *Renewal*

Name ____________________

Company ____________________

Address ____________________

City: __________ State: __________ Zip: __________ Country: __________

Phone: __________ Fax: __________ Email: __________

MEMBER PROFILE: Please help us serve you better!

By answering the following questions, you provide us with information that will allow us to plan our activities to meet your needs. All information is entirely confidential.

What is your job function?

1. ☐ System/Network Administrator
2. ☐ Consultant
3. ☐ Academic/Researcher
4. ☐ Developer/Programmer/Architect
5. ☐ System Engineer
6. ☐ Technical Manager
7. ☐ Student
8. ☐ Webmaster
9. ☐ Security

What is your role in the purchase decision

1. ☐ Final
2. ☐ Specify
3. ☐ Recommend
4. ☐ Influence
5. ☐ No role

☐ I do not want USENIX to email me notices of Association activities.

☐ I do not want my address made available for commercial mailings

PAYMENT OPTIONS:

Total enclosed $ __________

☐ I enclose a check/money order made payable to USENIX Association

☐ Enclosed is our purchase order (Educational, Corporate, and Supporting members only, please).

☐ Charge my: ☐ Visa ☐ MasterCard ☐ American Express

Account # ____________________ Expiration Date __________

Name on Card ____________________

Signature ____________________

Outside the USA? Please make your payment in US dollars by check drawn on US Bank, Visa/MasterCard, International postal money order

USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, California 94710 • Phone: 510 528 8649 • FAX: 510 548 5738 • Email: office@usenix.org

revised 3/14/00

CONNECT WITH USENIX & SAGE

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITES

http://www.usenix.org

http://www.sage.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

Send email to jel@usenix.org

CONTRIBUTIONS SOLICITED

Your are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of ;login: are 5 April 2000 and 2 May 2000.

USENIX & SAGE

The Advanced Computing Systems Association & The System Administrators Guild

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send address changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES